REMARKS

With entry of the foregoing amendment claims 1-3, 8-13, 18-20 and 22-31 remain in the application. Claims 4-7, 14-17 and 21 have been cancelled.

Claims 1, 11, 19 and 28-31 now have been amended. Support for these amendments to independent claims 1, 11 and 19 can be found in the specification and drawings as originally filed. For example, Figs. 3 and 4 of the specification and the accompanying text at pages 8 through 11 explain how differentiated storage pools are created by assigning different RAID configuration levels to storage device locations that are grouped according to performance variations. Fig. 4 depicts a specific example of an extent based (or page-based) allocation operation where the differentiated storage pools are of the type depicted in Fig. 3. In a particular example, the allocation operation provides a first pool A of the available storage to support a first RAID level 10. A second pool B of storage subspace is employed to support a RAID 5 service level, and a subspace C is configured to support a RAID 50 performance level.

No new matter has been added by entry of these amendments to the claims.

Claim Objections

Claims 28-31 were objected to as lacking antecedent basis for the recitation "the system of claim 11." The foregoing amendment eliminates the use of system and refers to a process. It is believed that this objection should now be withdrawn.

Specification Objections

The specification still stands objected to because it claims priority to U.S. Provisional Application Serial Number 60/441,810, filed January 21, 2003. Applicants maintain that there is indeed adequate support provided for the claims as they presently stand when the entire contents of the referenced provisional application are considered.

Due to apparent confusion on the part of the Examiner as to the content of the provisional, Applicants are providing as Attachment A to this response a complete copy of the

entire submission of U.S. Provisional Application Serial Number 60/441,810 as filed on January 21, 2003[1].

The contents of the specification as filed on that date are seen to have included a total of 64 pages. As per the "Provisional Application for Patent Cover Sheet", the application consisted of the cover sheet, a Fee Transmittal followed by 25 pages of specification, 11 sheets of drawings, and a Technical Appendix of 26 pages.

The Examiner's reason for denying the priority claim appears to be based on a supposed difference in attorney docket numbers. The document actual does have several different attorney docket numbers. For example, "EQLC-P60-003" is typed on the Cover Sheet and in the header of the first 25 pages of text. Other docket numbers such as "EQLC-PXX-007" appear on later pages, for example on page 39 entitled "Distributed Snapshot". Still other docket numbers such as "EQLC-PXX-006" appear on a page entitled "Client Load Distribution" at page 44, and yet a further docket number "EQLC-PXX-005" is used at page 54. But this is simply an artifact of the provisional application filing having included a combination of several documents.

The numbering that is consistent on all pages is the U.S. Patent Office's own serial number and date stamp of "60441810.012103". This appears exactly the same on the upper right hand corner of every single one of the 64 pages. The fact that certain attorney document numbers appear or do not appear on the document pages is not dispositive of the issue of whether the Applicant is entitled to claim priority.

The content of the provisional application as filed on January 21, 2003 does indeed enable the invention. We draw the Examiner's attention to the Technical Appendix portion and in particular, to the last 11 pages thereof. Excerpts from pages including the 54th and 60th through 64th pages, are reproduced again for the convenience of the Examiner in portions of Exhibits B through G herewith.

More particularly, we note that the drawings comprising these pages of the provisional filing correspond identically to the drawings in the present application. For example, the 61st page contains a drawing entitled "Diagram 1 – Storage Device Performance Differences by LBN" also labeled "Fig. 6". That drawing corresponds identically (with the exception of

[1] The copy in Attachment A was downloaded from the Public PAIR website at uspto.gov by the undersigned attorney on February 5, 2009. It thus represents the Patent Office's own record of what was included with the provisional application filed on January 21, 2003.

reference numerals) to Fig. 1 of the present application as filed on January 20, 2004. Please refer to Exhibit B, which has on the left side a reproduction of the 61st page from the provisional and on the right side a reproduction of Fig. 1 of the present utility application as filed.

There is similar identical correspondence between the 62nd through 64th pages (labeled Figs. 7, 8 and 9) of the provisional application and Figs. 2, 3 and 4 of the present utility application. See Exhibits C, D and E. Likewise, Fig. 5 of the provisional compares to Fig. 5 of the present utility application. See Exhibit F.

Thus, U.S. Provisional Application No. 60/441,810 contained several drawings that illustrate how differentiated pools of storage offer different RAID level performance (RAID 10 and RAID 50) across their LBN space. But there is even more support found in the text. The following text is an exact quote from the 54th page of the provisional (a copy of that page is reproduced in Exhibit G hereto):

> In a block storage environment with multiple devices, differentiated pools of storage are created by applying multiple RAID techniques simultaneously on each device. These pools of storage differ in regards to performance and reliability. Adaptive load balancing can be used to place data in pools to optimize service for a single or multiple consumers of storage services. It is stated there, for example:
>
> Storage devices offer different performance across their LBN space: some LBN ranges have higher bandwidth for read and/or write, some LBN ranges have higher throughput for small and/or random operations, some LBN ranges have slower bandwidth, and some LBN ranges have slower throughput.
>
> RAID levels have different performance characteristics for bandwidth and throughput for read and write operations, as well as in both normal mode and degraded operating modes.
>
> Device performance differences across LBN space, and performance and reliability differences of RAID techniques, can be combined of to create storage pools that offer different classes of service across a common set of storage devices to the consumers of storage services.
>
> Adaptive load balancing provides the ability to adjust data placement in pools for continuous optimal performance.

The technology to be protected is – minimally – :

The use of multiple raid levels concurrently on each device to create differentiated classes of storage service.

The use of multiple raid levels concurrently on each device in combination with device LBN performance differences to create differentiated classes of storage service.

Thus, the text of the provisional also described pooled storage having differentiated performance levels (A, B, C) assigned to respective different RAID configurations (RAID 10 and RAID 50).

Accordingly, withdrawal of the objection to the claim of priority to U.S. Provisional Application No. 60/441,810 is again respectfully requested.

The Claims as Amended are not Anticipated by Dimitri

Previous claims 1-4, 8-14, 18, and 22-31 were rejected under 35 U.S.C. 102(e) as being anticipated by Dimitri (U.S. Patent 6,839,802). With entry of the foregoing amendment, claims 1, 11, and 19 now more particularly recite that the system or process has several differentiated performance LBN subspaces which respectively support a different one of several RAID levels. This feature not found in the prior art.

As one example of the invention described in the application, a single RAID system provides different levels of RAID storage functionality for different classes, or "pools" of storage. A measurement system scans individual devices (such as a depicted storage device 10 of Fig. 1 of the application), and measures certain characteristics to determine how the device should be subdivided. Using the results of this measurement process, a determination is made that a collection of logical block names (LBNs) can be aggregated. The aggregated blocks having a common performance characteristics then provide a subspace within the LBN space. See page 7 line 9 through page 8 line 9 of the utility application as originally filed.

As known to one that is skilled in the art, there are different types of RAID systems in general. For example the original U.C. Berkeley "RAID" documents defined six (6) RAID

levels. Each RAID level defines a different way to spread data across multiple available disk drives in a system. For example, RAID level 0 does not provide redundancy and splits data across different disk drives. Since no redundant information is stored, the performance is very good, but the failure of any disk results in data loss. RAID level 0 is also commonly referred to as "striping".

RAID level 1 is commonly referred to as "mirroring" and uses two hard drives. This level provides redundancy by duplicating the data from one drive onto another drive. The performance of RAID level 1 is slightly better than RAID 0, since if a single fails, no data is lost.

RAID level 10 (such as assigned to the depicted pool A22 in Fig. 2 of the present application) employs a dual level array that uses multiple RAID 1 (mirrored disks) into a single array. Data is striped across all mirrored sets.

RAID 50 (as employed by pool B24 in the present application) is a dual level array that employs multiple RAID 5 levels into a single array.

Applicants' system 20 depicted in Fig. 2 employs multiple storage devices to support two different classes of storage (in a general sense) and specifically supports two different RAID levels. In this particular example, the same storage system is supporting both RAID level 10 with pool A22 and RAID level 50 with pool B24.

Fig. 4 and the accompanying text at page 10 line 11 through page 11 line 5 illustrate this advantage. For example, a given volume may have two areas that are (1) a high activity area assigned to pool A and other areas that are (2) low activity areas (highlighted in white) that are assigned to another pool C. Pool A areas may have high activity thus maybe assigned to a RAID 10 array. The RAID 10 array provides higher performance for higher activity but lower redundancy. At the same time, low activity areas assigned to pool C can be used to support a RAID 50 configuration. These slower performing areas can provide higher redundancy.

As a result, multiple instances of a RAID level storage service can be created using the underlying performance differentiated storage pools.

Prior to the Applicants' invention, systems such as that shown in Dimitri would have required different physical devices to support different RAID level types. Applicants do note that Dimitri does provide a system that enables differentiated "classes" of storage and organizes these according to performance levels. (These are called "zones", since Dimitri's disk is a zone

constant angular velocity ("VCAV") formatted disk.) A VCAV disk media inherently spins at a constant rotation per minute such that inner zones inherently have a greater data rate per revolution than outer zones. While it is not clear how logical block addresses (LBAs) map to "zones" in Dimitri, it is clear that his system will select a zone according to a set of performance criteria and treat them similarly. For example, at Fig. 4 and column 8 lines 31 through 43 of Dimitri as relied upon by the Examiner, Dimitri does teach that a zone would be selected based on a combination of file size and utilization history of the file. After selecting a zone, the zone control unit 50 then writes all data stripes to the same zone on the different disks.

The Dimitri architecture thus avoids a situation where overall performance of a RAID array is limited to the performance of the inner most zone of the disks. However, Dimitri only mentions RAID generally and does not recognize the possibility of different RAID levels at all in the same storage system. In fact, all that Dimitri ever suggests using is a type of disk stripping which equates to RAID level 0. (Dimitri at column 8 lines 22 through 25.)

Especially with entry of the foregoing amendment, Dimitri does not provide or even suggest a way to support different RAID levels on the same disk system. For this reason alone, claim 1 should be allowed.

In other aspects of the present invention, a performance process measures the performance of storing locations by making experimental read and write operations across the logical block main space, and uses these measurements to determine whether various locations can be aggregated to regions. (See for example, Applicants' specification at page 7, lines 4-11). A mapping process thus aggregates the logical block names for location having an identical level of performance (i.e., the partition locations). This results in the creation of different storage pools. The Applicants' claimed system clearly assigns different RAID levels to these different regions based upon the determined levels of performance of the locations within the regions (see at least the Applicants' specification at page 2, lines 17 through 19, and page 4 lines 9 through 11, and Fig. 4). Clients thus accessing such a system can utilize the storage pool and select from a number of different classes of RAID level service from the very same storage array (see the specification at page 11, lines 5 through 8). For example, one client may utilize a RAID 10 level service, while another client may utilize a RAID 5 level service, both however storing their data

on the same set of physical disks (see the specification at page 9, line 22 through page 10 line 2). This is not taught or suggested by Dimitri.

Claims 1, 9 and 11 as amended, all require that the multiple differentiated (aggregated) storage regions be assigned different RAID level configurations.

All other claims depend from claims 1, 9 or 11. Thus, all claims should be considered to be patentable over Dimitri.

Information Disclosure Statement

An Information Disclosure Statement (IDS) is being filed concurrently herewith. Entry of the IDS is respectfully requested.

CONCLUSION

In view of the above amendments and remarks, it is believed that all claims are in condition for allowance, and it is respectfully requested that the application be passed to issue. If the Examiner feels that a telephone conference would expedite prosecution of this case, the Examiner is invited to call the undersigned.

Respectfully submitted,

HAMILTON, BROOK, SMITH & REYNOLDS, P.C.

By _____

David J. Thibodeau, Jr.
Registration No. 31,671
Telephone: (978) 341-0036
Facsimile: (978) 341-0136

Concord, MA 01742-9133
Date: 2/6/09

J1032 U.S. PTO
01/21/03

01-23-05

PTO/SB/16 (10-01)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

jc715 U.S. PTO
60/441810
01/21/03

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Express Mail Label No. EV 161060204 US

INVENTOR(S)		
Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
G. Paul	Koning	Brookline, New Hampshire
Peter C.	Hayden	Mount Vernon, New Hampshire
Paula	Long	Hollis, New Hampshire

[ ] *Additional inventors are being named on the _____ separately numbered sheets attached hereto*

TITLE OF THE INVENTION (500 characters max)

Block Data Migration

Direct all correspondence to: **CORRESPONDENCE ADDRESS**

[x] Customer Number [ ] → *Customer Number*

OR

* 2 8 1 2 0 *
Customer Number Bar Code

[x] Firm or Individual Name	Paul E. Lewkowicz ROPES & GRAY				
Address	One International Place				
City	Boston	State	MA	Zip	02110-2624
Country	US	Telephone	(617) 951-7000	Fax	(617) 951-7050

ENCLOSED APPLICATION PARTS (*check all that apply*)

[x] Specification *Number of Pages* 25 — [ ] CD(s), Number
[x] Drawing(s) *Number of Sheets* 11 — [x] Other: Technical Appendix (26 pages)
[ ] Application Data Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT

[x] Applicant claims small entity status. See 37 CFR 1.27.
[ ] A check or money order is enclosed to cover the filing fees
[x] The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 18-1945

FILING FEE AMOUNT ($): 80.00

[ ] Payment by credit card. Form PTO-2038 is attached.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

[x] No [ ] Yes, the name of the U.S. Government agency and the Government contract number are:

Respectfully submitted,

SIGNATURE Paul E. Lewkowicz

Date January 21, 2003

TYPED OR PRINTED NAME Paul E. Lewkowicz

REGISTRATION NO. *(if appropriate)* 44,870

TELEPHONE (617) 951-7173

Docket Number: EQLC-P60-003

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Airbill No. EV 161060204 US, in an envelope addressed to: Box Provisional Patent Application, Commissioner for Patents, Washington, DC 20231, on the date shown below.

Dated: 1/21/03 Signature: Ginny Blundell (Ginny Blundell)

9036793_1

EXHIBIT
tabbies
A

PTO/SB/17 (10-02)
Approved for use through 10/31/2002. OMB 0651-0032
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2003

Patent fees are subject to annual revision.

[X] Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 80.00

Complete if Known	
Application Number	Not Yet Assigned
Filing Date	January 21, 2003
First Named Inventor	G. Paul Koning
Examiner Name	Not Yet Assigned
Group Art Unit	N/A
Attorney Docket No.	EQLC-P60-003

METHOD OF PAYMENT *(check all that apply)*

[ ] Check [ ] Credit Card [ ] Money Order [ ] Other [ ] None

[X] Deposit Account

Deposit Account Number: 18-1945

Deposit Account Name: Ropes & Gray

The Commissioner is hereby authorized to: *(check all that apply)*

[X] Charge fee(s) indicated below [X] Credit any overpayments

[X] Charge any additional fee(s) during the pendency of this application

[ ] Charge fee(s) indicated below, except for the filing fee

to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	750	2001	375	Utility filing fee	
1002	330	2002	165	Design filing fee	
1003	520	2003	260	Plant filing fee	
1004	750	2004	375	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	80.00

SUBTOTAL (1) ($) 80.00

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

		Extra Claims	Fee from below	Fee Paid
Total Claims	-20** =		x =	
Independent Claims	-3** =		x =	
Multiple Dependent			=	

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	84	2201	42	Independent claims in excess of 3
1203	280	2203	140	Multiple dependent claim, if not paid
1204	84	2204	42	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0.00

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge – late filing fee or oath	
1052	50	2052	25	Surcharge – late provisional filing fee or cover sheet.	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	410	2252	205	Extension for reply within second month	
1253	930	2253	465	Extension for reply within third month	
1254	1,450	2254	725	Extension for reply within fourth month	
1255	1,970	2255	985	Extension for reply within fifth month	
1401	320	2401	160	Notice of Appeal	
1402	320	2402	160	Filing a brief in support of an appeal	
1403	280	2403	140	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive – unavoidable	
1453	1,300	2453	650	Petition to revive - unintentional	
1501	1,300	2501	650	Utility issue fee (or reissue)	
1502	470	2502	235	Design issue fee	
1503	630	2503	315	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	750	2809	375	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	750	2810	375	For each additional invention to be examined (37CFR 1.129(b))	
1801	750	2801	375	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	
Other fee (specify)					

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 0.00

SUBMITTED BY				Complete *(if applicable)*	
Name (Print/Type)	Paul E. Lewkowicz	Registration No. (Attorney/Agent)	44,870	Telephone	(617) 951-7173
Signature	Paul E. Lewkowicz			Date	January 21, 2003

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail, Airbill No. EV 161060204 US, in an envelope addressed to: Box Provisional Patent Application, Commissioner for Patents, Washington, DC 20231, on the date shown below.

Dated: 1/21/03 Signature: (Ginny Blundell)

9036795_1

"Express Mail" mailing label number:

EV 161 060 204 US

PROVISIONAL APPLICATION

BLOCK DATA MIGRATION

G. Paul KONING
Peter C. HAYDEN
Paula LONG

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to systems and methods for data storage in computer networks, and more particularly to systems that store data resources across a plurality of servers.

Description of the Related Art

The client server architecture has been one of the more successful innovations in information technology. The client server architecture allows a plurality of clients to access services and data resources maintained and/or controlled by a server. The server listens for requests from the clients and in response to the request determines whether or not the request can be satisfied, responding to the client as appropriate. A typical example of a client server system has a "file server" set up to store data files and a number of clients that can communicate with the server. Clients typically request that the server to grant access to different ones of the data files maintained by the file server. If a data file is available and a client is authorized to access that data file, the server can deliver the requested data file to the server and thereby satisfy the client's request.

Although the client server architecture has worked remarkably well, it does have some drawbacks. For example, the number of clients contacting a server and the number of requests being made by individual clients can vary significantly over time. As such, a server responding to client requests may find itself inundated with a volume of requests that is impossible or nearly impossible to satisfy. To address this problem, network administrators

often make sure that the server includes sufficient data processing assets to respond to anticipated peak levels of client requests. Thus, for example, the network administrator may make sure that the server comprises a sufficient number of central processing units (CPUs) with sufficient memory and storage space to handle the volume of client traffic that may arrive.

Note that, in this disclosure, the term "resource" is employed to describe the files, data blocks or pages, applications, or other services or capabilities provided by the server to clients. The term "asset" is employed to describe the processing hardware, memory, storage devices, and other elements available to the server for the purpose of responding to client requests.

Even with a studied determination of needed system resources, variations in client load can still burden a server or group of servers acting in concert as a system. For example, even if sufficient hardware assets are provided in the server system, it may be the case that client requests focus on a particular file, data block within a file, or other resource maintained by the server. Thus, continuing with the above example, it is not uncommon that client requests overwhelmingly focus on a small portion of the data files maintained by the file server. Accordingly, even though the file server may have sufficient hardware assets to respond to a certain volume of client requests, if these requests are focused on a particular resource, such as a particular data file, most of the file server assets will remain idle while those assets that support the data file being targeted are over-burdened.

To address this problem, network engineers have developed load balancing systems that distribute client requests across the available assets for the purpose of distributing client demand on individual assets. To this end, the load balancing system may distribute client requests in a round-robin fashion that evenly distributes requests across the available server assets. In other practices, the network administrator sets up a replication system that can identify when a particular resource is the subject of a flurry of client requests and duplicate the targeted resource so that more of the server assets are employed in supporting client requests for that resource.

Furthermore, while servers do a good job of storing data, their assets are limited. One common technique employed today to extend server assets is to rely on peripheral storage devices such as tape libraries, RAID disks, and optical storage systems. When properly connected to servers, these storage devices are effective for backing up data online

and storing large amounts of information. By connecting a number of such devices to a server, a network administrator can create a "server farm" (comprised of multiple server devices and attached storage devices) that can store a substantial amount of data. Such attached storage devices are collectively referred to as Network Attached Storage (NAS) systems.

But as server farms increase in size, and as companies rely more heavily on data-intensive applications such as multimedia, this traditional storage model is not quite as useful. This is because access to these peripheral devices can be slow, and it is not always possible for every user to easily and transparently access each storage device.

In order to address this shortfall, a number of vendors have been developing an architecture called a Storage Area Network (SAN). SANs provide more options for network storage, including much faster access to NAS-type peripheral devices. SANs further provide flexibility to create separate networks to handle large volumes of data.

A SAN is a high-speed special-purpose network or sub-network that interconnects different kinds of data storage devices with associated data servers on behalf of a larger network of users. Typically, a storage area network is part of the overall network of computing assets of an enterprise. SANs support disk mirroring, backup and restore, archiving, and retrieval of archived data, data migration from one storage device to another, and the sharing of data among different servers in a network. SANs can incorporate sub-networks containing NAS systems.

A SAN is usually clustered in close proximity to other computing resources (such as mainframes) but may also extend to remote locations for backup and archival storage, using wide area networking technologies such as asynchronous transfer mode (ATM) or Synchronous Optical Networks (SONET). A SAN can use existing communication technology such as optical fiber ESCON or Fibre Channel technology to connect storage peripherals and servers.

Although SANs hold much promise, they face a significant challenge. Bluntly, consumers expect a lot of their data storage systems. Specifically, consumers demand that SANs provide network-level scalability, service, and flexibility, while at the same time providing data access at speeds that compete with server farms.

This can be quite a challenge, particularly in multi-server environments, where a client wishing to access specific information or a specific file is redirected to a server that has the piece of the requested information or file. The client then establishes a new connection to the other server upon redirect and severs the connection to the originally contacted server. However, this approach defeats the benefit of maintaining a long-lived connection between the client and the initial server.

Another approach is "storage virtualization" or "storage partitioning" where an intermediary device is placed between the client and a set of physical (or even logical) servers, with the intermediary device providing request routing. None of the servers are aware that it is providing only a portion of the entire partitioned service, nor are any of the clients aware that the data resources are stored across multiple servers. Obviously, adding such an intermediary device adds complexity to the system.

Although the above techniques may work well in certain client server architectures, they each require additional devices or software (or both) disposed between the clients and the server assets to balance loads by coordinating client requests and data movement. As such, this central transaction point can act as a bottleneck that slows the server's response to client requests.

Furthermore, resources must be supplied continuously, in response to client requests, with strictly minimized latency. Accordingly, there is a need in the art for a method for rapidly distributing client load across a server system while at the same time providing suitable response times for incoming client resource requests and preserving a long-lived connection between the client and the initial server.

SUMMARY OF THE INVENTION

The systems and methods described herein include systems for managing responses to requests from a plurality of clients for access to a set of resources. In one embodiment, the systems comprise a plurality of equivalent servers wherein the set of resources is partitioned across this plurality of servers. Each equivalent server has a load monitor process that is capable of communicating with the other load monitor processes for generating a measure of the client load on the server system and the client load on each of the respective servers. The system further comprises a resource distribution process that is

responsive to the measured system load and is capable of repartitioning the set of resources to thereby redistribute the client load.

In another embodiment, the systems and methods described herein include storage area network systems (SANs) that may be employed for providing storage assets for an enterprise. The SAN of the invention comprises a plurality of servers and/or network devices. At least a portion of the servers and network devices include a load monitor process that monitors the client load being placed on the respective server or network device. The load monitor process is further capable of communicating with other load monitor processes operating on the storage area network. Each load monitor process may be capable of generating a system-wide load analysis that indicates the client load being placed on the storage area network. Additionally, the load monitor process may be capable of generating an analysis of the client load being placed on that respective server and/or network device. Based on the client load information observed by the load monitor process, the storage area network is capable of redistributing client load to achieve greater responsiveness to client requests. To this end, in one embodiment, the storage area network is capable of repartitioning the stored resources in order to redistribute client load.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects and advantages of the invention will be appreciated more fully from the following further description thereof, with reference to the accompanying drawings, wherein:

FIGURE 1 depicts schematically the structure of a prior art system for providing access to a resource maintained on a storage area network;

FIGURE 2 presents a function block diagram of one system according to the invention;

FIGURE 3 presents in more detail the system depicted in FIGURE 2;

FIGURE 4 depicts the flow of data through layers of a network;

FIGURE 5 depicts in more detail one embodiment of a system according to the invention;

FIGURE 6 presents a flow chart diagram of one process according to the invention;

FIGURE 7 is a schematic diagram of a client server architecture with servers organized in server groups;

FIGURE 8 is a schematic diagram of the server groups as seen by a client;

FIGURE 9 shows details of the information flow between the client and the servers of a group;

FIGURE 10 is a process flow diagram for retrieving resources in a partitioned resource environment;

FIGURE 11 depicts in more detail and as a functional block diagram one embodiment of a system according to the invention; and

FIGURE 12 depicts an example of a routing table suitable for use with the system of FIGURE 7.

The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION OF THE ILLUSTRATED EMBODIMENTS

To provide an overall understanding of the invention, certain illustrative embodiments will now be described, including a system that provides a storage area network (SAN) that more efficiently responds to client load changes by migrating data blocks while providing continuous data access. However, it will be understood by one of ordinary skill in the art that the systems and methods described herein can be adapted and modified to redistribute resources in other applications, such as distributed file systems, database applications, and/or other applications where resources are partitioned or distributed. Moreover, such other additions and modifications fall within the scope hereof will not and do not depart from the scope of the invention.

FIGURE 1 depicts a prior art network system for supporting requests for resources from a plurality of clients 12 that are communicating across a local area network 24. Specifically, FIGURE 1 depicts a plurality of clients 12, a local area network (LAN) 24, and a storage system 14 that includes a switch 16, a master data table 18, and a plurality of servers 22a-22n. The storage system 14 may provide a storage area network (SAN) that provide storage resources to the clients 12 operating across the LAN 24. As further shown in FIGURE 1, each client 12 may make a request 20 for a resource maintained on the SAN

14. Each request 20 is delivered to the switch 16 and processed therein. During processing is that the clients 12 can request resources across the LAN 24 and during processing, the switch 16 employs the master data table 18 to identify which of the plurality of servers 22a through 22n has the resource being requested by the respective client 12.

In FIGURE 1, the master data table 18 is depicted as a database system, however in alternative embodiments the switch 16 may employ a flat file master data table that it maintained by the switch 16. In either case, the switch 16 employs the master data table 18 to determine which of the servers 22a through 22n maintains which resources. Accordingly, the master data table 18 acts as an index that lists the different resources maintained by the system 14 and which of the underlying servers 22a through 22n is responsible for which of the resources.

As further depicted by FIGURE 1, once the switch 16 determines the appropriate server 22a through 22n for the requested resource, the retrieved resource may be passed from the identified server through the switch 16 and back to the LAN 24 for delivery (represented by arrow 21) to the appropriate client 12. Accordingly, FIGURE 1 depicts that system 14 employs the switch 16 as a central gateway through which all requests from the LAN 24 are processed. The consequence of this central gateway architecture is that delivery time of resources requested by clients 12 from system 14 can be relatively long and this delivery time may increase as latency periods grow due to increased demand for resources maintained by system 14.

Turning to FIGURE 2, a system 10 according to the invention is depicted. Specifically, FIGURE 2 depicts a plurality of clients 12, a local area network (LAN) 24, and a server group 30 that includes plurality of servers 32A through 32N. As shown by FIGURE 2, the clients 12 communicate across the LAN 24. As further shown in FIGURE 2, each client 12 may make a request for a resource maintained by server group 30. In one application, the server group 30 is a storage area network (SAN) that provides network storage resources for clients 12. Accordingly, a client 12 may make a request across the (LAN) 24 that is transmitted, as depicted in FIGURE 2 as request 34, to a server, such as the depicted server 32B of the SAN 30.

The depicted SAN 30 comprises a plurality of equivalent servers 32A through 32N. Each of these servers has a separate IP address and thus the system 10 appears as a storage

area network that includes a plurality of different IP addresses, each of which may be employed by the clients 12 for accessing storage resources maintained by the SAN 30.

The depicted SAN 30 employs the plurality of servers 32A though 32N to partition resources across the storage area network, forming a partitioned resource set. Thus, each of the individual servers may be responsible for a portion of the resources maintained by the SAN 30. In operation, the client request 34 received by the server 32B is processed by the server 32B to determine the resource of interest to that client 12 and to determine which of the plurality of servers 32A through 32N is responsible for that particular resource. In the example depicted in FIGURE 2, the SAN 30 determines that the server 32A is responsible for the resource identified in the client request 34. As further shown by FIGURE 2, the SAN 30 employs a system where, rather than have the original server 32B respond to the client request 34, a shortcut response is employed that allows the responsible server to respond directly to the requesting client 12. Server 32A thus delivers response 38 over LAN 24 to the requesting client 12.

As discussed above, the SAN 30 depicted in FIGURE 2 comprises a plurality of equivalent servers. Equivalent servers will be understood, although not limited to, server systems that expose a uniform interface to a client or clients, such as the clients 12. Each equivalent server will respond in the same manner to a request presented by any client 12. Thus, since each server 32A through 32N presents the same response to any client 12, it is immaterial to the client 12 which of the servers 32A through 32N responds to its request.

Each of the depicted servers 32A through 32N may comprise conventional computer hardware platforms such as one of the commercially available server systems from the Sun Microsystems Inc. of Santa Clara, California. Each server executes one or more software processes for the purpose of implementing the storage area network. The SAN 30 may employ a Fibre Channel network, an arbitrated loop, or any other type of network system suitable for providing a storage area network. As further shown in FIGURE 2, each server may maintain its own storage resources or may have one or more additional storage devices coupled to it. These storage devices may include, but are not limited to, RAID systems, tape library systems, disk arrays, or any other device suitable for providing storage resources to the clients 12.

It will be understood that those of ordinary skill in the art that the systems and methods of the invention are not limited to storage area network applications and may be

applied to other applications where it may be more efficient for a first server to receive a request and a second server to generate and send a response to that request. Other applications may include distributed file systems, database applications, application service provider applications, or any other application that may benefit from this short-cut response technique.

FIGURES 4 and 5 depict in more detail one embodiment of a short-cut response suitable for use with the system depicted in FIGURE 2. Specifically, FIGURE 4 depicts a functional block diagram that shows one example of how connection and state information is created during a short-cut response operation. As will be discussed in more detail hereinafter, when a client 12 and a server 32 exchange information, a connection is established between the client and the server. For each established connection, the server 32 maintains some information about the connection that the server will employ when responding to the client 12, such as (for example) the client's IP address. In a traditional client server exchange, the server that received the client request also responds to that request. In these cases, then, the server that responds to the request is also the server that set up the connection to the client 12. Thus, all of the connection information is available at the server when the server needs that information to make its response.

Under the short-cut response process described herein, the server generating and transmitting the response may be different from the server that received the request and set up the connection. Accordingly, the server generating the response may need to access the connection information residing on the receiving server in order to be able to directly respond to the client 12.

On example of a short-cut response is depicted in FIGURE 4. Specifically, FIGURE 4 depicts two servers 32A and 32B that take part in a short-cut response to a request 34 transmitted from one of the clients 12. In this embodiment, the receiving server is server 32A and the responding server is server 32B. The short-cut response technique described herein allows the request 34 to be received by receiving server 32A. The receiving server 32A may determine that server 32B is actually responsible for the resource that has been identified in the request 34. Accordingly, the receiving server 32A may forward the request 34 to the responding server 32B. The responding server 32B may determine the appropriate response to the request 34. The responding server 32B may then prepare a response 38 that may be transmitted directly to the client 12, without passing through server 32A. However, as discussed above, to generate the response 38, the

responding server 32B accesses connection information generated by the receiving server 32A in response to the request 34 transmitted by the client.

As depicted in FIGURE 4, when the client request 34 is received by the receiving server 32A, the client request may be handled by a plurality of network layers including the physical layer 46. Such a layer model is described with reference to the well-known Open Systems Interconnect (OSI) seven layer model. Typically the physical layer employs an Ethernet adapter card and the well-known Ethernet protocol; however, the physical layer employed by the storage area network 10 can vary according to the application in ways well-known to one of ordinary skill in the art.

The request 34 may be processed by the physical layer 46 and may then be processed by the data link layer 48 and network layer 50. In the depicted embodiment, layers 48 and 50 together employ the Internet Protocol (IP) layer and the Transmission Control Protocol (TCP), commonly referred to as TCP/IP communications. Other protocols known in the art are equally adaptable for use with embodiments of the systems and methods described herein.

As further shown in FIGURE 4, both at the IP layer and the TCP layer, connection information 54 and 58 (respectively) are established. This information can include information representative of the IP address of the client 12 that generated request 34 and to which the response 38 is to be forwarded. At the TCP layer, connection information 58 may be established that includes information such as the number of datagrams or packets received or other kinds of similar information.

After processing at the TCP layer 50, the request 34 may be further processed at higher protocol layers 52, e.g., in on or more of the transport, session, or application layers defined in the OSI reference model. In the depicted embodiment, at least one of the upper layers of the model may be provided by the Internet Small Computer Serial Interface (ISCSI) protocol and thus employed as part of the storage area network. At this higher protocol layer 52, connection information 60 may be stored. Connection information 60 is representative of the upper level connection information relevant to the proper functioning of the ISCSI application program. In addition to connection information, it will also be understood that state information, such as HTTP cookies and other similar information may be maintained and stored.

In either case, it will be seen that at different levels of the network protocol employed by the SAN 30, information is generated that is relevant to generating a response to the client 12. In the shortcut response process described herein, the connection and state information 54, 58, and 60 maintained at the different layers of the protocol stack are shared with the responding server 32B. As shown in FIGURE 4, the server 32B has a similarly established protocol stack. The protocol stack includes a physical layer 62, data link layer 64, transport layer 68, and high level layer 70 (in one embodiment of the present invention, ISCSI). Similarly, as in the receiving server 32A, each of the layers 64, 68 and 70 provides for storing connection or state information, such as the depicted connection and state information, represented by functional blocks 72, 74 and 78 respectively.

In responding to the request 34, the receiving server 32A travels through the network stack, passing through each layer. As is known to those of ordinary skill in the art, request information at each layer of the stack is processed by unpacking information, reviewing header information in the request, and performing other functions. Such other functions include, but are not limited to, setting up connections and related information that may be employed to respond to the request 34.

As further shown in FIGURE 4, in the upper layer 52, the ISCSI protocol may determine that the server 32B is actually responsible for the resource requested by the client 12. Accordingly, in this application the ISCSI protocol may forward the client's request to the server 32B.

Upper protocol layers 60 on server 32A forward the client request to the upper protocol layers 70 of server 32B, which is (in this example) in control of the resource requested by the client 12. Server 32A may include a distributed socket server process that is capable of providing access to the connection and state information 54, 58, and 60 maintained at the server 32A. As depicted in FIGURE 4, the result of the distributed socket server process is to provide the server 32B with access to this connection and state information as if the request 34 had been received and routed through the protocol stack in server 32B. As shown by path 79 in FIGURE 4, the distributed socket server process achieves an effect as if the request 34 had been routed from server 32A to server 32B. Consequently, the server 32B will obtain the necessary connection and state information to generate the response 38 for the client 12.

Turning to FIGURE 5, one embodiment of a distributed socket server process is depicted. Specifically, FIGURE 5 depicts the equivalent servers 32A and 32B. Server 32A has a plurality of layers in the protocol stack that include layers 46, 48, 50 and 52. Along side the layers is shown the distributed socket server 80. Similarly, the server 32B is shown as having a plurality of layers in the protocol stack including layers 62, 64, 68, and 70. Server 32B has a distributed socket server process 82 that is shown as being in communication with each of the layers of the protocol stack.

Also depicted in FIGURE 5 is an example of a data exchange between the socket server 80 of Server 32A and the application layer 70 and socket server 82 of the server 32B. As shown in FIGURE 5, the distributed socket server 80 may work with a socket established by the application (or other, higher level) layer 52. In one embodiment, the application 52 receives the request 34 (not shown) from the client 12 (not shown) and determines that server 32A is not the server responsible for the resource requested by the client. The server 32A then determines or identifies the server in the SAN that is responsible for the requested resource. In FIGURE 5, the application layer 52 determines that server 32B is responsible for the requested resource. The application layer 52 then forwards the request to the application level 70 operating on server 32B. The forwarding of the request is depicted by the communication 84 shown on FIGURE 5. The application layer 70 receives the forwarded request and processes it. Upon processing the request, the server 32B determines that the connection and state information for generating the response 38 (not shown) is stored at server 32A. Accordingly, the application layer 70 can direct the socket server 82 to request connection and state information from the socket server 80.

The socket server 80 is in communication with each layer of the network stack. Accordingly, the socket server 80 can gather the appropriate connection and state information from the different layers of the protocol stack and transfer the collected connection and state information to the socket server 82 operating on server 32B. The socket server 82B can store or establish the appropriate information at the appropriate network layers 64, 68 and 70. Accordingly, the distributed socket server 82 configures the server 32B to generate the response 38.

Turning to FIGURE 6, one process for short-cut response is shown as a flow chart diagram. FIGURE 6 depicts a process 90 that begins in a step 92 when a client such as one of the depicted clients 12 sends an encapsulated request to the storage area network (SAN). After step 92, the process 90 proceeds to step 94 wherein the request passes through the

protocol stack of the server to which the client directed the request. The request, while passing through the protocol stack, alters and creates connection information and state information related to this request. At block 98, the process 90 determines the appropriate server for responding to the request generated by the client 12. In those cases where the responding server is different from the server that received the request, the process 90 may forward, otherwise route, or redirect the request to the identified (responding) server through means well-known in the art.

At block 100, the responding server may process the forwarded request to determine the server that has the connection and state information necessary for generating the short-cut response. In an alternative practice, the forwarded request may also contain the connection state information necessary to enable the identified responding server, such as server 32B, to generate the appropriate response to the client. In either case, the server that has been identified as being responsible for the resource requested by the client now has the request from the client as well as the connection state information necessary to respond to that request. In block 102, that server can create the response and send it through the layers of the protocol stack and on to the client. The distributed socket server can then update the connection and state information on the appropriate server (block 103) and in the process may terminate, block 104.

The above description, with reference to FIGURES 1 through 6, discloses a short cut response system and method, that is one embodiment, employs a socket server that executes as a process on each of the servers in the SAN 30. As shown, the short cut response methods disclosed herein may be employed as part of a client server architecture where the server that receives a client request may be different from the server that responds to the request. These methods have been described with reference to a SAN application, but this is only for the purpose of clarity and the methods disclosed herein are suitable for use in a wide variety of applications.

Referring now to FIGURE 7, one or several clients 12 are connected, for example via a network 24, such as the Internet, an intranet, a WAN or LAN, or by direct connection, to servers 161, 162, and 163 that are part of a server group 116.

As described above, the depicted clients 12 can be any suitable computer system such as a PC workstation, a handheld computing device, a wireless communication device,

or any other such device, equipped with a network client program capable of accessing and interacting with the server group 116 to exchange information with the server group 116.

Servers 161, 162 and 163 employed by the system 110 may be conventional, commercially available server hardware platforms, as described above. However any suitable data processing platform may be employed. Moreover, it will be understood that one or more of the servers 161, 162, or 163 may comprise a network device, such as a tape library, or other device, that is networked with the other servers and clients through network 24.

Each server 161, 162, and 163 may include software components for carrying out the operation and the transactions described herein, and the software architecture of the servers 161, 162, and 163 may vary according to the application. In certain embodiments, the servers 161, 162, and 163 may employ a software architecture that builds certain of the processes described below into the server's operating system, into device drivers, into application level programs, or into a software process that operates on a peripheral device (such as a tape library, RAID storage system, or another storage device or any combination thereof). In any case, it will be understood by those of ordinary skill in the art that the systems and methods described herein may be realized through many different embodiments and that the particular embodiment and practice employed will vary as a function of the application of interest. All these embodiments and practices accordingly fall within the scope of the present invention.

In operation, the clients 12 will have need of the resources partitioned across the server group 116. Accordingly, each of the clients 12 will send requests to the server group 116. The clients 12 typically act independently, and as such, the client load placed on the server group 116 will vary over time. In a typical operation, a client 12 will contact one of the servers, for example server 161, to access a resource, such as a data block, page (comprising a plurality of blocks), file, database table, application, or other resource. The contacted server 161 itself may not hold or have control over the requested resource. However, in a preferred embodiment, the server group 116 is configured to make all the partitioned resources available to the client 12 regardless of the server that initially receives the request. For illustration, FIGURE 7 shows two resources, one resource 180 that is partitioned over all three servers (servers 161, 162, 163) and another resource 170 that is partitioned over two of the three servers. In the exemplary application of the system 110

being a block data storage system, each resource 170 and 180 may represent a partitioned block data volume.

The depicted server group 116 therefore provides a block data storage service that may operate as a storage area network (SAN) comprised of a plurality of equivalent servers, servers 161, 162, and 163. Each of the servers 161, 162, and 163 may support one or more portions of the partitioned block data volumes 170 and 180. In the depicted server group 116, there are two data resources (e.g., volumes) and three servers; however there is no specific limit on the number of servers. Similarly, there is no specific limit on the number of resources or data volumes. Moreover, each resource may be contained entirely on a single server, or it may be partitioned over several servers, either all of the servers in the server group, or a subset of the server group.

In practice, there may of course be limits due to implementation considerations, for example the amount of memory assets available in the servers 161, 162 and 163 or the computational limitations of the servers 161, 162 and 163. Moreover, the grouping itself, i.e., deciding which servers will comprise a group, may in one practice involve an administrative decision. In a typical scenario, a group might at first contain only a few servers, perhaps only one. The system administrator would add servers to a group as needed to obtain the level of performance required. Increasing servers creates more space (memory, disk storage) for resources that are stored, more CPU processing capacity to act on the client requests, and more network capacity (network interfaces) to carry the requests and responses from and to the clients. It will be appreciated by those of skill in the art that the systems described herein are readily scaled to address increased client demands by adding additional servers into the group 116. However, as client load varies, the server group 116 can redistribute client load to take better advantage of the available assets in server group 116.

To this end, the server group 116, in one embodiment, comprises a plurality of equivalent servers. Each equivalent server supports a portion of the resources partitioned over the server group 116. As client requests are delivered to the equivalent servers, the equivalent servers coordinate among themselves to generate a measure of system load and to generate a measure of the client load of each of the equivalent servers. In a preferred practice, this coordinating is done in a manner that is transparent to the clients 12, so that the clients 12.

Referring now to FIGURE 8, a client 12 connecting to a server 161 (FIGURE 7) will see the server group 116 as if the group were a single server having multiple IP addresses. The client 12 is not necessarily aware that the server group 116 is constructed out of a potentially large number of servers 161, 162, 163, nor is it aware of the partitioning of the block data volumes 170 and 180 over the several servers. A particular client 12 may have access to only a single server, through its unique IP address. As a result, the number of servers and the manner in which resources are partitioned among the servers may be changed without affecting the network environment seen by the client 12.

FIGURE 9 shows the resource 180 of FIGURE 8 as being partitioned across servers 161, 162 and 163. In the partitioned server group 116, any data volume may be spread over any number of servers within the server group 116. As seen in FIGURES 7 and 8, one volume 170 (Resource 1) may be spread over servers 162, 163, whereas another volume 180 (Resource 2) may be spread over servers 161, 162, 163. Advantageously, the respective volumes may be arranged in fixed-size groups of blocks, also referred to as "pages," wherein an exemplary page contains 8192 blocks. Other suitable page sizes may be employed, and pages comprising variable numbers of blocks (rather than fixed) are also possible.

In an exemplary embodiment, each server in the group 116 contains a routing table 165 for each volume, with the routing table 165 identifying the server on which a specific page of a specific volume can be found. For example, when the server 161 receives a request from a client 12 for volume 3, block 93847, the server 161 calculates the page number (page 11 in this example for the page size of 8192) and looks up in the routing table 165 the location or number of the server that contains page 11. If server 163 contains page 11, the request is forwarded to server 163, which reads the data and returns the data to the server 161. Server 161 then sends the requested data to the client 12. The response may be returned to the client 12 via the same server 161 that received the request from the client 12. Alternatively, the short-cut approach described above may be used.

Accordingly, it is immaterial to the client 12 as to which server 161, 162, 163 has the resource of interest to the client 12. As described above, the servers 162, 162 and 163 will employ the routing tables to service the client request, and the client 12 need not know ahead of time which server is associated with the requested resource. This allows portions of the resource to exist at different servers. It also allows resources, or portions thereof, to be moved while the client 12 is connected to the partitioned server group 116. This latter

type of resource re-partitioning is referred to herein as "block data migration" in the case of moving parts of resources consisting of data blocks or pages. One of ordinary skill in the art will of course see that resource parts consisting of other types of resources (discussed elsewhere in this disclosure) may also be moved by similar means. Accordingly, the invention is not limited to any particular type of resource.

Data may be moved upon command of an administrator or automatically by storage load balancing mechanisms such as those discussed herein. Typically, such movement or migration of data resources is done in groups of blocks referred to as pages.

When a page is moved from one equivalent server to another, it is important for all of the data, including that in the page being moved, to be continuously accessible to the clients so as not to introduce or increase response time latency. In the case of manual moves, as implemented in some servers seen today, the manual migration interrupts service to the clients. As this is generally considered unacceptable, automatic moves that do not result in service interruptions are preferable. In such automatic migrations, the movement must needs be transparent to the clients.

According to one embodiment of the present invention, a page to be migrated is considered initially "owned" by the originating server (i.e., the server on [on in] which the data is initially stored) while the move is in progress. Routing of client read requests continue to go through this originating server.

Requests to write new data into the target page are handled specially: data is written to both the page location at the originating server and to the new (copy) page location at the destination server. In this way, a consistent image of the page will end up at the destination server even if multiple write requests are processed during the move.

A more elaborate approach may be used when pages become large. In such cases, the migration may be done in pieces: a write to a piece that has already been moved is simply redirected to the destination server; a write to a piece currently being moved goes to both servers as before. Obviously, a write to a piece not yet moved may be processed by the originating server.

Such write processing approaches are necessary to support the actions required if a failure should occur during the move, such as a power outage. If the page is moved as a single unit, an aborted (failed) write can begin over again from the beginning. If the page is

moved in pieces, the move can be restarted from the piece that was in transit at the failure. It is the possibility of restart that makes it necessary to write data to both the originating and destination servers.

Table 1 shows the sequence of block data migration stages for a unit block data move from a server A to Server B; Table 2 shows the same information for a piece-wise block data move.

Table 1

Stage	Restart Stage	Destination	Action	Read	Write
1	N/A	Server A	Not started	Server A	Server A
2	2	Server A	Start move	Server A	Servers A and B
3	2	Server A	Finish Move	Server A	Servers A and B
4	N/A	Server B	Routing Table updated	Server B	Server B

Table 2

Stage	Restart Stage	Destination	Action	Read	Write
1	N/A	Server A	Not started	Server A	Server A
2	2		Start move, piece 1	Server A	Servers A and B for piece 1, server A for others
3	2	Server A	Finish move, piece 1	Server B for piece 1, server A for others	Server B for piece 1, server A for others
4	4		Start move, piece 2	Server B for piece 1, server A for others	Server B for piece 1, servers A and B for piece 2, server A for others
5	4		Finish move, piece 2	Server B for pieces 1 and 2, server A for others	Server B for pieces 1 and 2, server A for others
...	(repeat as needed for all pieces)				
n	N/A	Server B	Routing Table updated	Server B	Server B

Upon moving a resource, the routing tables 165 (referring again to FIGURE 9) are updated as necessary (through means well-known in the art) and subsequent client requests

will be forwarded to the server now responsible for handling that request. Note that, at least among servers containing the same resource 170 or 180, the routing tables 165 may be identical.

In some embodiments of the present invention, once the routing tables are updated, the page at the originating server (or "source" resource) is deleted by standard means. Additionally, a flag or other marker is set in the originating server for the originating page location to denote, at least temporarily, that that data is no longer valid. Any latent read or write requests still destined for the originating server will thus trigger an error and subsequent retry, rather than reading the expired data on that server. By the time any such retries return, they will encounter the updated routing tables and be appropriately directed to the destination server. In no case are duplicated, replicated, or shadowed copies of the block data (as those terms are known in the art) left on in the server group.

FIGURE 10 depicts an exemplary client allocation process 400 for handling client requests in a partitioned server group 116. The client allocation process 400 begins at 410 by receiving a request for a resource, such as a file or blocks of a file, at 420. The client allocation process 40 checks, in operation 430, if the requested resource is present at the initial server that received the request from the client 12. If the requested resource is present at the initial server, the initial server returns the requested resource to the client 12, at 480, and the process 400 terminates at 490. Conversely, if the requested resource is not present at the initial server, the server will consult a routing table, operation 440, to determine which server actually holds the resource requested by the client, operation 450. The request may then be forwarded to the server that holds the requested resource, operation 460, which may return the requested resource to the initial server, operation 480. (Alternatively, the short-cut process described above with respect to FIGURE 6 may be employed) The process 400 then goes to 480 as before, to have the initial server forward the requested resource to the client 12, and the process 400 terminates at 490.

Accordingly, one of ordinary skill in the act will see that the system and methods described herein are capable of migrating one or more partitioned resources over a plurality of servers, thereby providing a server group capable of handling requests from multiple clients. The resources so migrated over the several servers can be directories, individual files within a directory, blocks within a file or any combination thereof. Other partitioned services may be realized. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned

server that supports applications being delivered over the Internet. In general, the approach can be applied to any service where a client request can be interpreted as a request for a piece of the total resource.

Turning now to FIGURE 11, one particular embodiment of the system 500 is depicted wherein the system is capable of redistributing client load to provide more efficient service. Specifically, FIGURE 11 depicts a system 500 wherein the clients 12A through 12E communicate with the server block 116. The server block 116 includes three equivalent servers, server 161, 162, and 163, in that each of the servers will provide substantially the same response to the same request from a client. As such, from the perspective of the clients 12, the server group 116 appears to be a single server system that provides multiple network or IP addresses for communicating with clients 12A-12E.

Each server includes a routing table, depicted as routing tables 200A, 200B and 200C, a load monitor process 220A, 220B and 220C respectively, a client allocation process 320A, 320B, and 320C, a client distribution process 300A, 30B, 300B and 300C and a resource transfer process, 240A, 240B and 240C respectively. Further, and for the purpose of illustration only, the FIGURE 11 represents the resources as pages of data 280 that may be transferred from one server to another server.)

As shown by arrows in FIGURE 11, each of the routing tables 200A, 200B, and 200C are capable of communicating with each other for the purpose of sharing information. As described above, the routing tables may track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 116. Because each of the equivalent servers 161, 162 and 163 are capable (by definition) of providing the same response to the same request from a client 12, routing tables 200A, 200B, and 200C (respectively) coordinate with each other to provide a global database of the different resources and the specific equivalent servers that are responsible for those resources.

FIGURE 12 depicts one example of a routing table 200A and the information stored therein.

As depicted in FIGURE 11, each routing table includes an identifier for each of the equivalent servers 161, 162 and 163 that support the partitioned data block storage group 116. Additionally, each of the routing tables includes a table that identifies those data blocks associated with each of the respective equivalent servers. In the routing table embodiment depicted by FIGURE 12, the equivalent servers support two partitioned

volumes. A first one of the volumes is distributed or partitioned across all three equivalent servers 161, 162, and 163. The second partitioned volume is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

FIGURE 11 illustrates pictorially that the system 500 may be capable of redistributing client load by having client 12C, which was originally communicating with server 161, redistributed to server 162. To this end, FIGURE 11 depicts an initial condition wherein the server 161 is communicating with clients 12A, 12B, and 12C. This is depicted by the bidirectional arrows coupling the server 161 to the respective clients 12A, 12B, and 12C. As further shown in FIGURE 11, in an initial condition, clients 12D and 12E are communicating with server 163 and no client (during the initial condition) is communicating with server 162. Accordingly, during this initial condition, server 161 is supporting requests from three clients, clients 12A, 12B, and 12C. Server 162 is not servicing or responding to requests from any of the clients.

Accordingly, in this initial condition the server group 116 may determine that server 161 is overly burdened or asset constrained. This determination may result from an analysis that server 161 is overly utilized given the assets it has available. For example, it could be that the server 161 has limited memory and that the requests being generated by clients 12A, 12B, and 12C have overburdened the memory assets available to server 161. Thus, server 161 may be responding to client requests at a level of performance that is below an acceptable limit. Alternatively, it may be determined that server 161, although performing and responding to client requests at an acceptable level, is overly burdened with respect to the client load (or bandwidth) being carried by server 162. Accordingly, the server group 116 may make a determination that overall efficiency may be improved by redistributing client load from its initial condition to one wherein server 162 services requests from client 12C.

To this end, FIGURE 11 depicts this redistribution of client load by illustrating a connection 325 (depicted by a dotted bi-directional arrow) between client 12C and server 162. It will be understood that after redistribution of the client load, the communication path between the client 12C and server 161 may terminate.

In operation, each of the depicted servers 161, 162, and 163 is capable of monitoring the complete load that is placed on the server group 116 as well as the load from each client and the individual client load that is being handled by each of the respective servers 161,

162 and 163. To this end, each of the servers 161, 162 and 163 include a load monitoring process 220A, 220B, and 220C respectively. As described above, the load monitor processes 220A, 220B and 220C are capable of communicating among each other. This is illustrated in FIGURE 11 by the bidirectional lines that couple the load monitor processes on the different servers 161, 162 and 163.

Each of the depicted load monitor processes may be software processes executing on the respective servers and monitoring the client requests being handled by the respective server. The load monitors may monitor the number of individual clients 12 being handled by the respective server, the number of requests being handled by each and all of the clients 12, and/or other information.

Accordingly, the load monitor process 220A is capable of generating information representative of the client load applied to the server 161 and is capable of corresponding with the load monitor 220B of server 162. In turn, the load monitor process 220B of server 162 may communicate with the load monitor process 220C of server 163, and load monitor process 220C may communicate with process 220A (not shown). By allowing for communication between the different load monitor processes 220A, 220B, and 220C, the load monitor processes may determine the system-wide load applied to the server group 116 by the clients 12.

FIGURE 11 further illustrates that server 161 as well as the other servers, includes a client distribution process 300A, 300B and 300C respectively. In one embodiment, the client distribution process 300A is capable of processing information generated by the load monitor process 220A and employing that information for the purpose of distributing client load among the servers of the server group 116. To this end, the client distribution process 300A may be software process executing on server 161 and capable of processing the system load and client load information generated by the load monitor process to determine whether server 161 is servicing a disproportionate share of the client requests being handled by server group 116.

Each client distribution process 300 is equivalent to each other client distribution process 300 and may be a separate process executing on each equivalent server 161, 162, and 163.

In one practice, the client distribution process 300A determines from the loan monitor process 220A whether the server 161 is servicing or responding to too many of

clients 12. Upon a determination that server 161 is overburdened, the client distribution process 300A may determine from the system load information whether there is another server in server group 116 that is more available to handle a portion of the clients 12 being serviced by server 161. In the depiction of this process illustrated by FIGURE 11, the client distribution process 300A determines that server 162 would be more appropriate for handling at least one of the clients 12 being supported by server 161. More particularly, the client distribution process 300A determines that client 12C is to be redistributed from server 161 to server 162.

In this example, the client 12C may be continually requesting access to the same resource. For example, FIGURE 5 shows that such a resource may be the page 280, maintained by the server 161. For the purpose of distributing client 12C to server 162, the client distribution process 300A can activate the block data migration process 350 (described above) that transfers page 280 from server 161 to server 162. Accordingly, in the embodiment depicted in FIGURE 11 the client distribution process 300A cooperates with the resource transfer process 350 to re-partition the resources in a manner that is more likely to cause client 12C to continually make requests to server 162 as opposed to server 161.

Once the resource 280 has been transferred to server 162, the routing table 200B can update itself (by standard means well-known in the art) and update the routing tables 200A and 200C accordingly, again by standard means well-known in the art. In this way, the resources may be repartitioned across the servers 161, 162 and 163 in a manner that redistribute client load as well.

An Exemplary Embodiment

One embodiment may consist of an apparatus for resource migration, comprising a storage system having a plurality of storage servers with a set of resources partitioned thereon, said storage servers each having a load monitor process capable of communicating with other load monitor processes for generating a measure of loading on respective ones of the plurality of servers; a resource migration process for transferring a resource from one of said plurality of servers to another of said plurality of servers in response to said measure of loading.

In this apparatus, said servers may be equivalent to each other. Alternatively, said resources may comprise one or more of the group consisting of data blocks, program files, multimedia files, applications, and database files. Or, said measure of loading may reflect

both a storage system load and a server load. Additionally, said storage system may be a Storage Area Network.

Alternate Embodiments

The order in which the steps of the present method are performed is purely illustrative in nature. In fact, the steps can be performed in any order or in parallel, unless otherwise indicated by the present disclosure.

The method of the present invention may be performed in either hardware, software, or any combination thereof, as those terms are currently known in the art. In particular, the present method may be carried out by software, firmware, or microcode operating on a computer or computers of any type. Additionally, software embodying the present invention may comprise computer instructions in any form (e.g., source code, object code, interpreted code, etc.) stored in any computer-readable medium (e.g., ROM, RAM, magnetic media, punched tape or card, compact disc (CD) in any form, DVD, etc.). Furthermore, such software may also be in the form of a computer data signal embodied in a carrier wave, such as that found within the well-known Web pages transferred among devices connected to the Internet. Accordingly, the present invention is not limited to any particular platform, unless specifically stated otherwise in the present disclosure.

Moreover, the depicted system and methods may be constructed from conventional hardware systems and specially developed hardware is not necessary. For example, in the depicted systems, the clients can be any suitable computer system such as a PC workstation, a handheld computing device, a wireless communication device, or any other such device equipped with a network client hardware and or software capable of accessing a network server and interacting with the server to exchange information. Optionally, the client and the server may rely on an unsecured communication path for accessing services on the remote server. To add security to such a communication path, the client and the server can employ a security system, such the Secured Socket Layer (SSL) security mechanism, which provides a trusted path between a client and a server. Alternatively, they may employ another conventional security system that has been developed to provide to the remote user a secured channel for transmitting data over a network.

Furthermore, networks employed in the systems herein disclosed may include conventional and unconventional computer to computer communications systems now known or engineered in the future, such as but not limited to the Internet.

Servers may be supported by a commercially available server platform such as a Sun Microsystems, Inc. Sparc ™ system running a version of the UNIX operating system and running a server program capable of connecting with, or exchanging data with, clients.

Those skilled in the art will know, or be able to ascertain using no more than routine experimentation, many equivalents to the embodiments and practices described herein.

As discussed above, the invention disclosed herein can be realized as a software component operating on a conventional data processing system such as a UNIX workstation. In that embodiment, the short cut response mechanism can be implemented as a C language computer program, or a computer program written in any high level language including C++, C#, Pascal, FORTRAN, Java, or BASIC. Additionally, in an embodiment where microcontrollers or digital signal processors (DSPs) are employed, the short cut response mechanism can be realized as a computer program written in microcode or written in a high level language and compiled down to microcode that can be executed on the platform employed. The development of such code is known to those of skill in the art, and such techniques are set forth in Digital Signal Processing Applications with the TMS320 Family, Volumes I, II, and III, Texas Instruments (1990). Additionally, general techniques for high level programming are known, and set forth in, for example, Stephen G. Kochan, Programming in C, Hayden Publishing (1983).

While particular embodiments of the present invention have been shown and described, it will be apparent to those skilled in the art that changes and modifications may be made without departing from this invention in its broader aspect and, therefore, the appended claims are to encompass within their scope all such changes and modifications as fall within the true spirit of this invention.

TECHNICAL APPENDICES

The following Appendices are hereby incorporated herein by reference in their entireties and form an integral part of this disclosure.

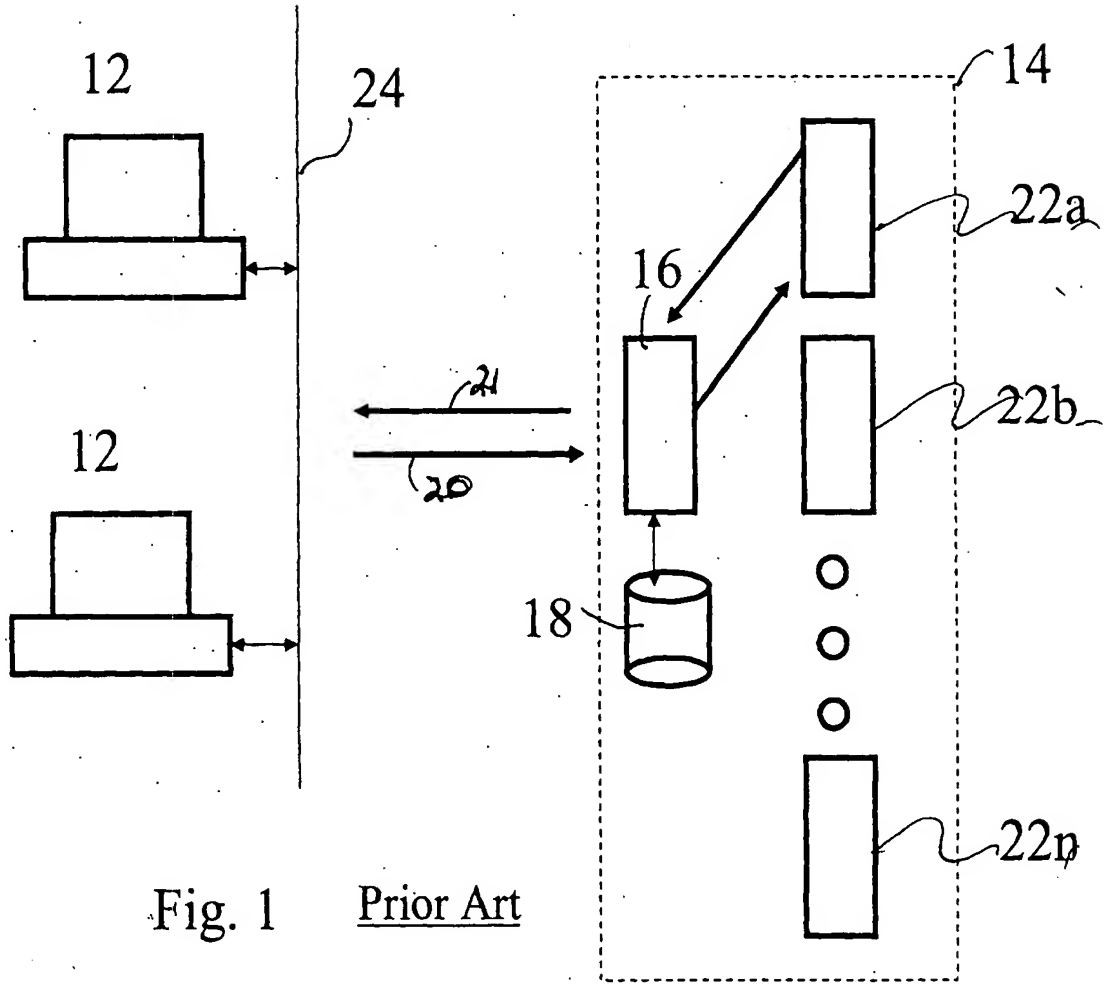


Fig. 1 Prior Art

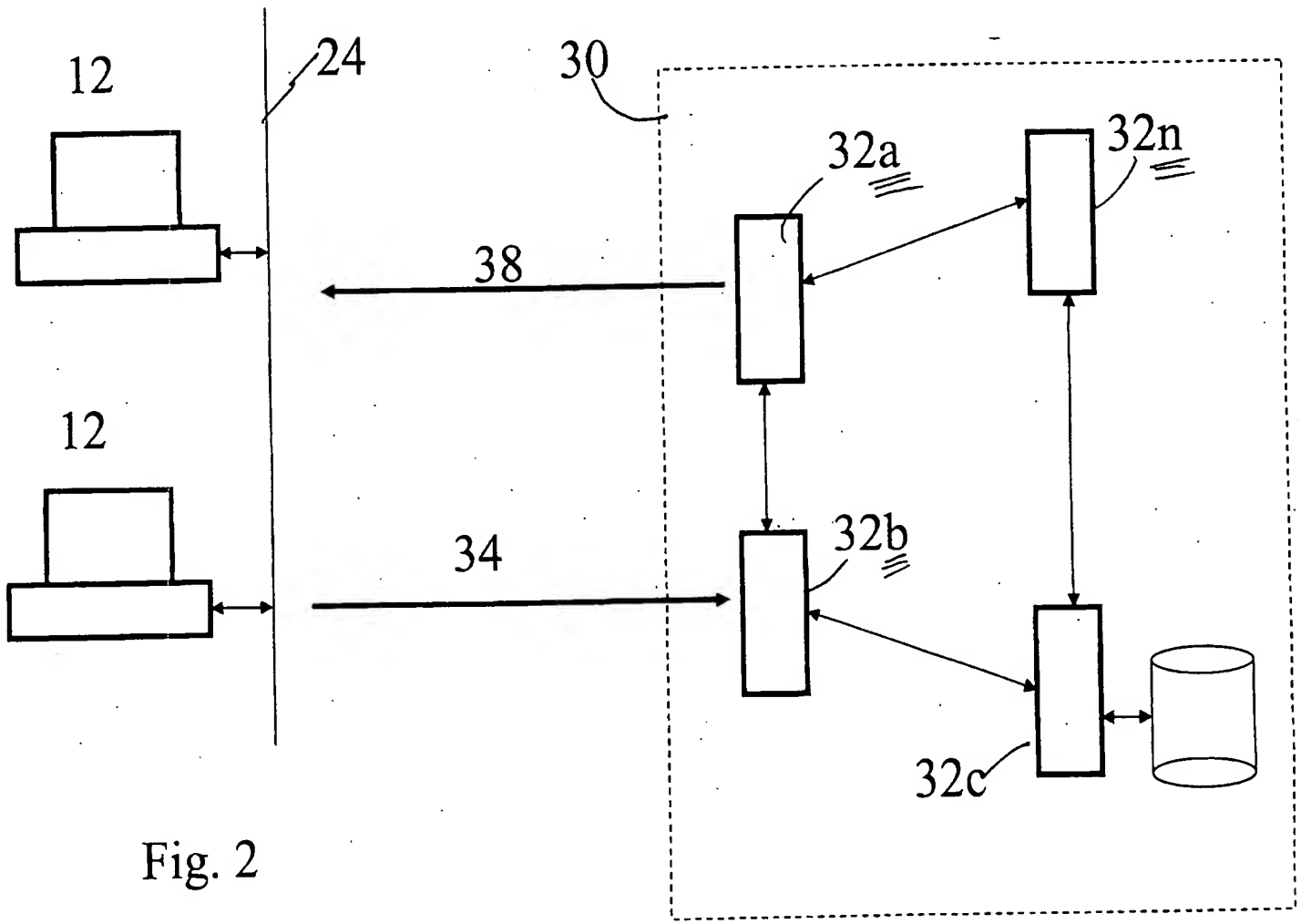


Fig. 2

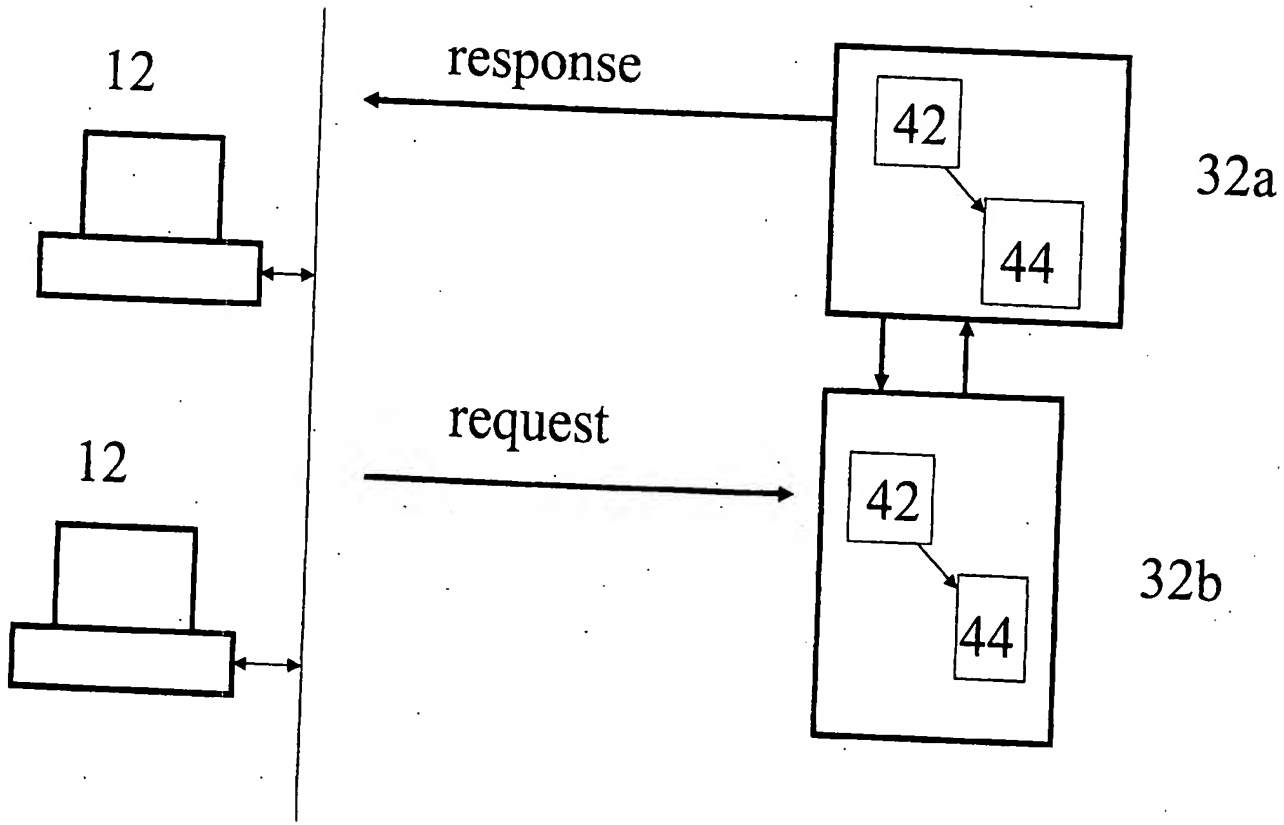


Fig. 3

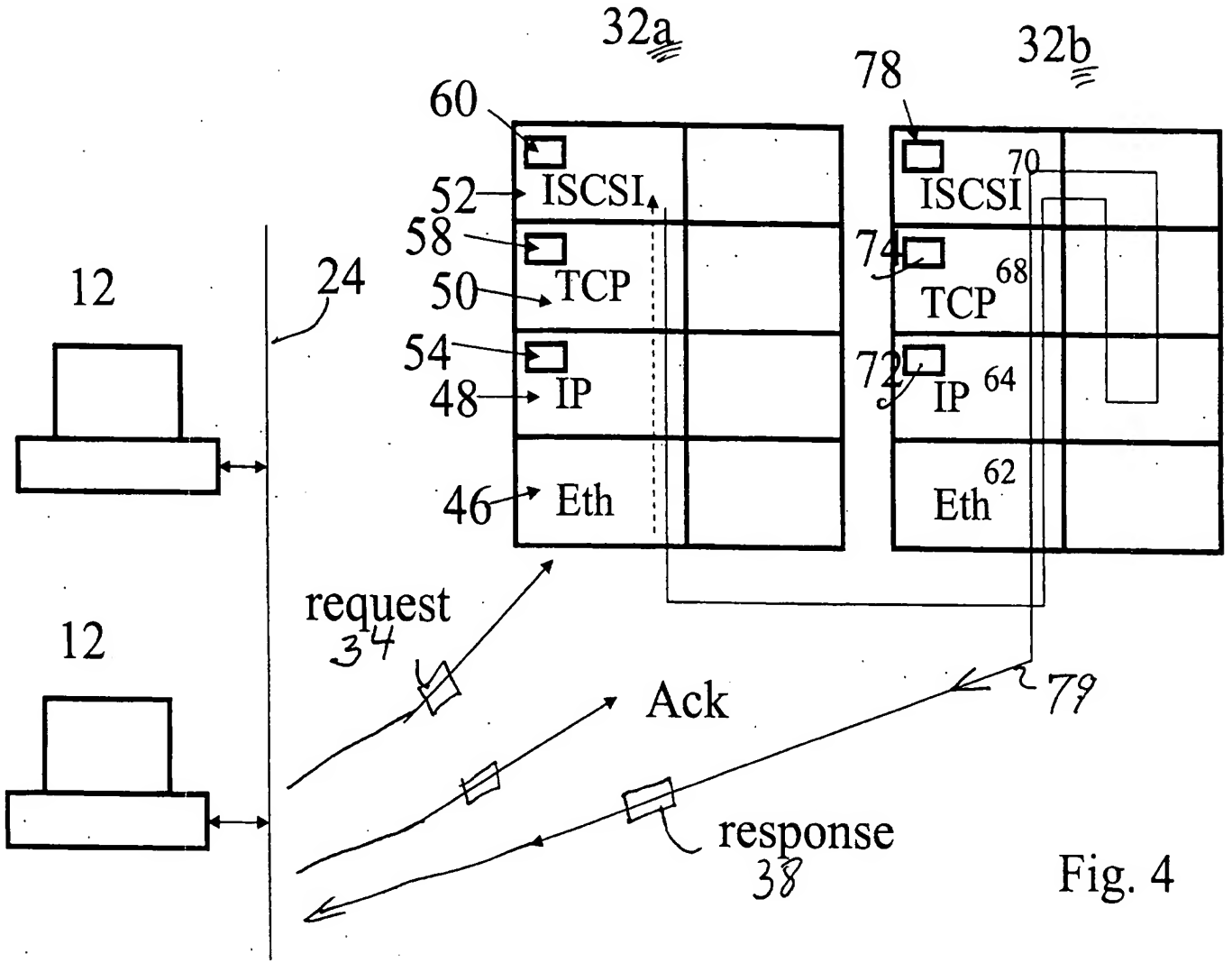

32a
32b
78
60
52
ISCSI
58
TCP
50
54
IP
48
46
Eth
70
ISCSI
74
TCP
68
72
IP
64
62
Eth
12
24
12
request
34
Ack
79
response
38

Fig. 4

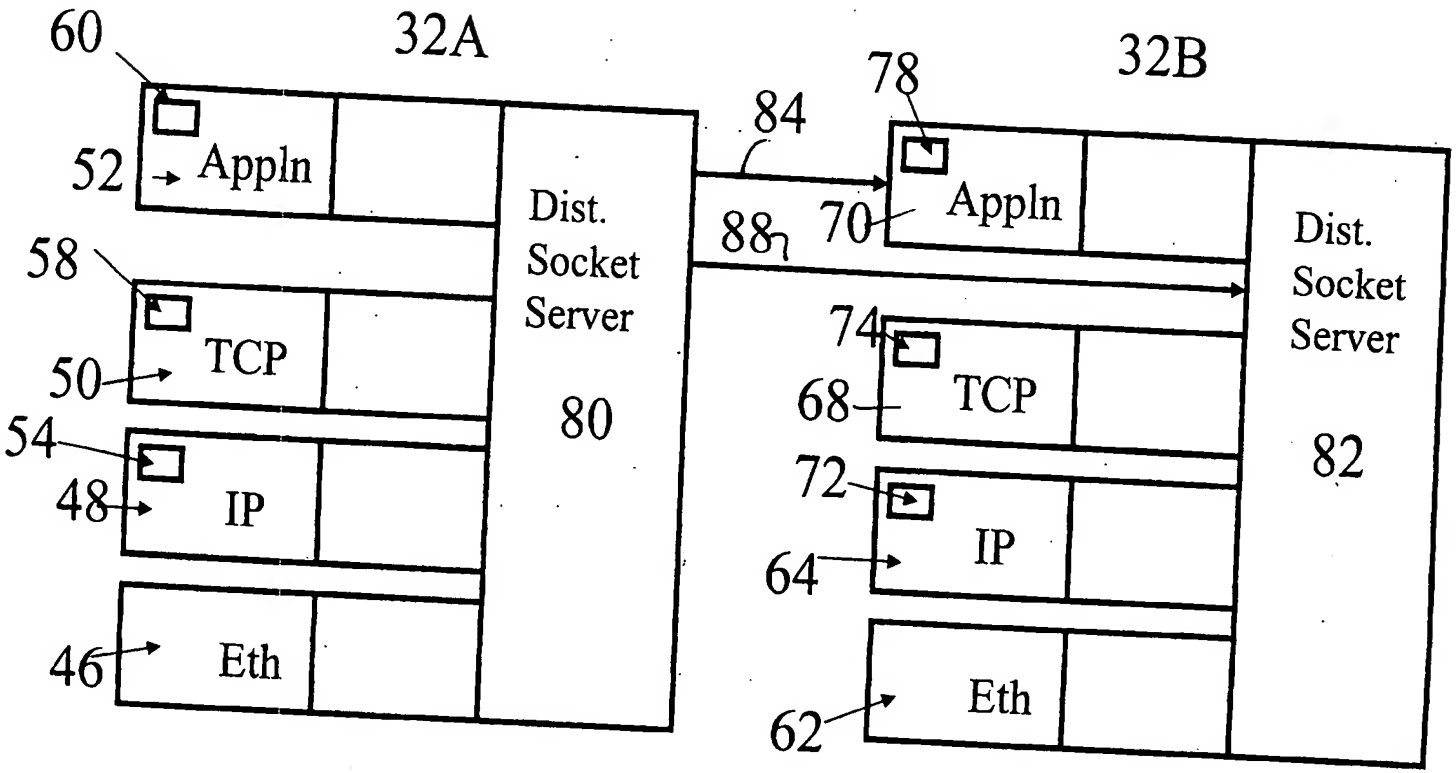

60
32A
32B
78
84
52
Appln
Dist.
Socket
Server
88
70
Appln
Dist.
Socket
Server
58
74
TCP
50
80
68
TCP
82
54
72
48
IP
IP
64
46
Eth
Eth
62

Fig. 5

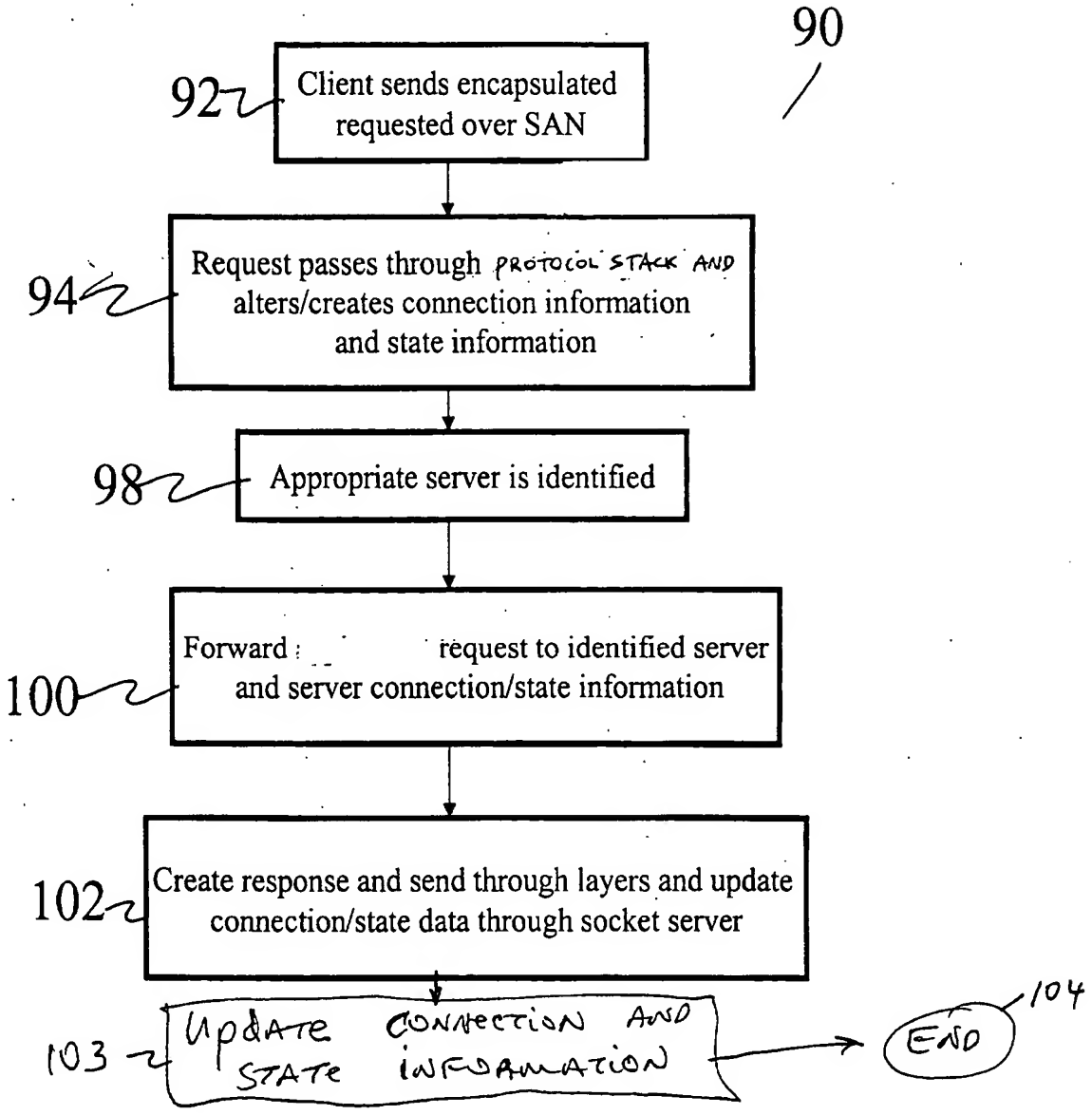

90
92
Client sends encapsulated requested over SAN
94
Request passes through PROTOCOL STACK AND alters/creates connection information and state information
98
Appropriate server is identified
100
Forward request to identified server and server connection/state information
102
Create response and send through layers and update connection/state data through socket server
103
Update CONNECTION AND STATE INFORMATION
END
104

Fig. 6

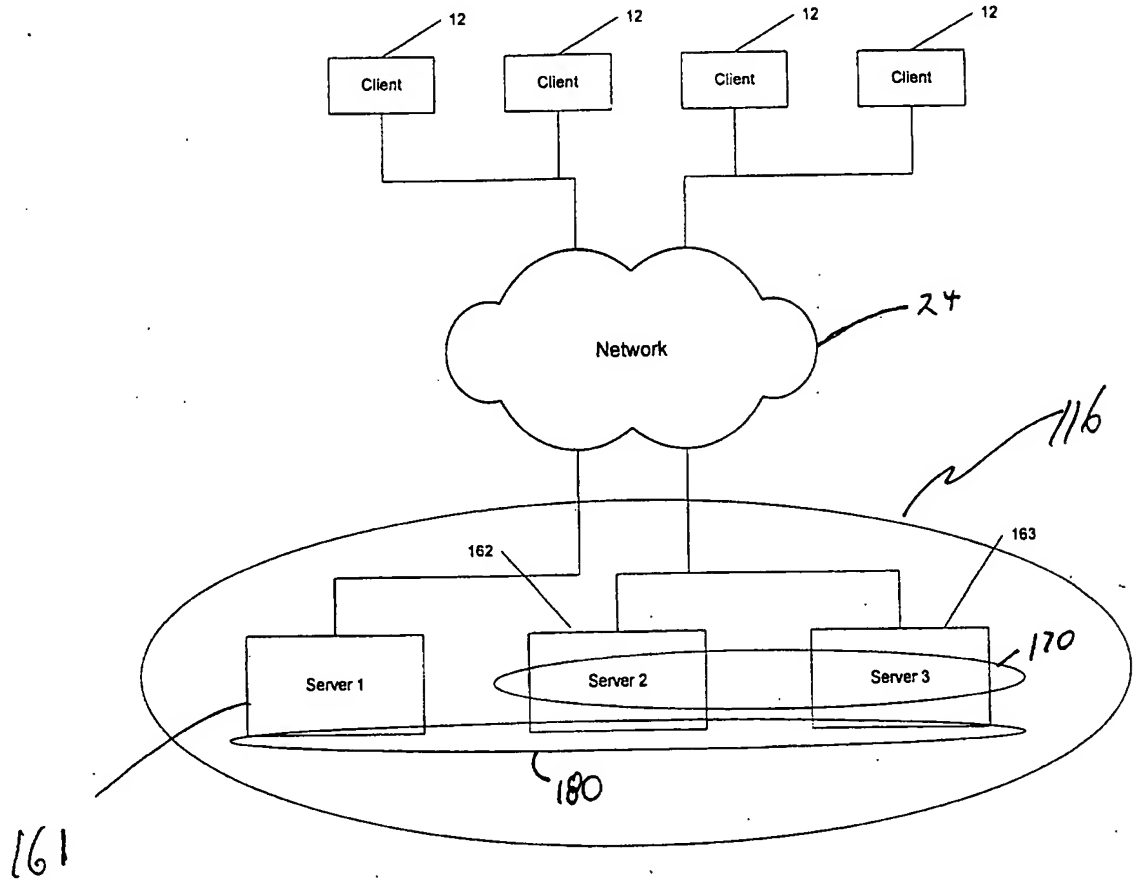

110
12
Client
12
Client
12
Client
12
Client
24
Network
116
162
163
Server 1
Server 2
Server 3
170
180
161

FIG. 7

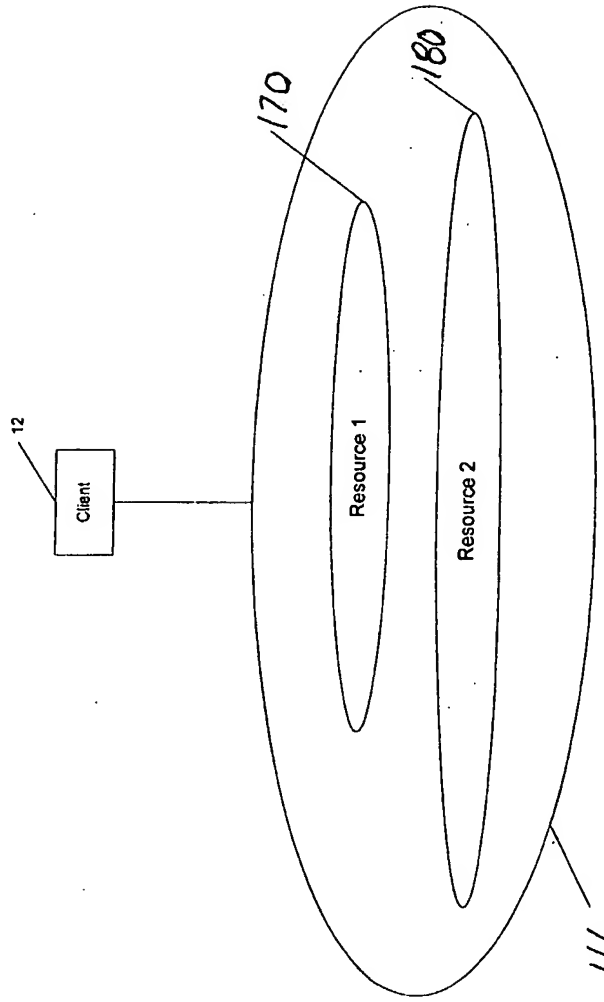


FIG. 8

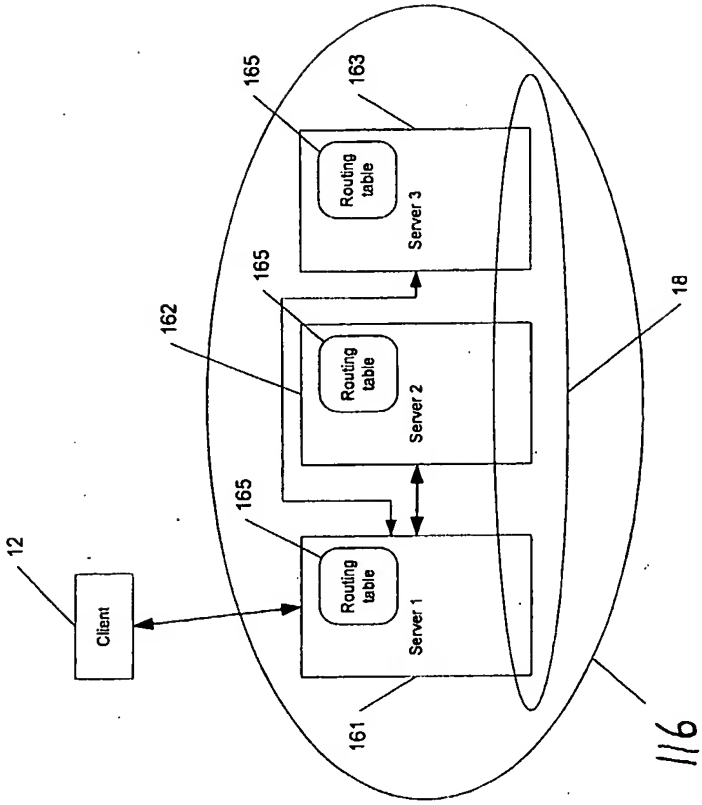

12
Client
Routing table
Server 1
161
165
162
Routing table
Server 2
165
Routing table
Server 3
163
165
18
116

FIG. 9

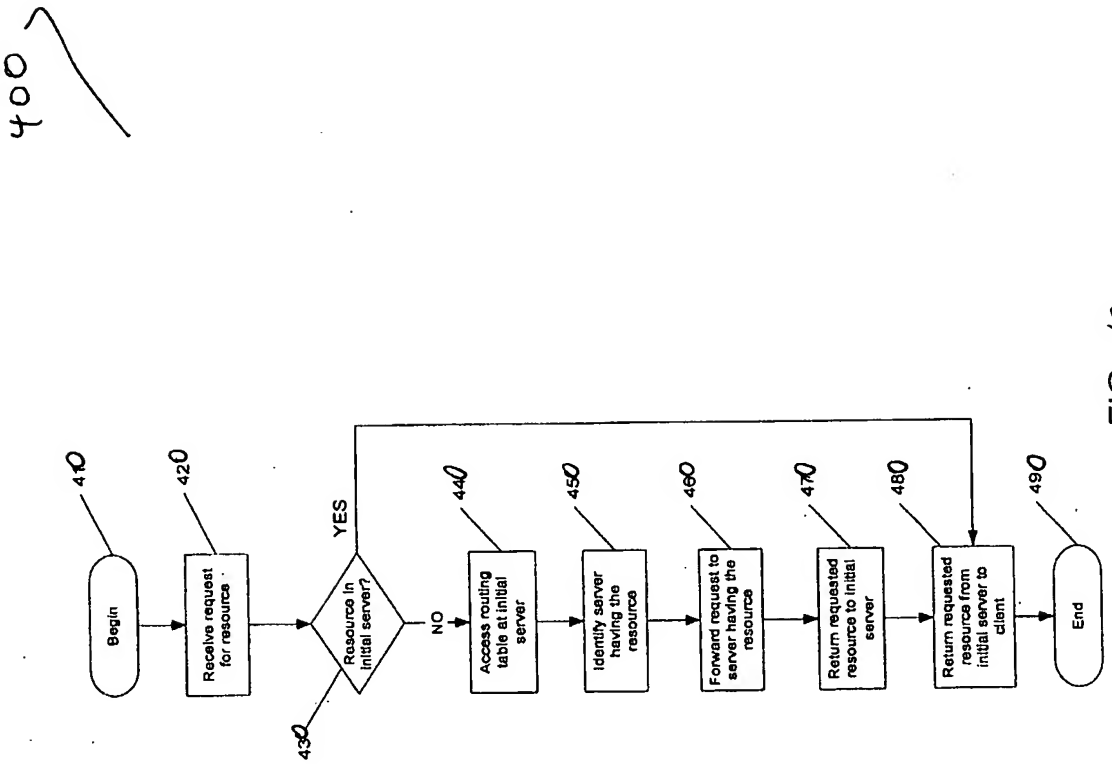

400
410
Begin
420
Receive request for resource
430
Resource in initial server?
YES
NO
440
Access routing table at initial server
450
Identify server having the resource
460
Forward request to server having the resource
470
Return requested resource to initial server
480
Return requested resource from initial server to client
490
End

FIG. 10

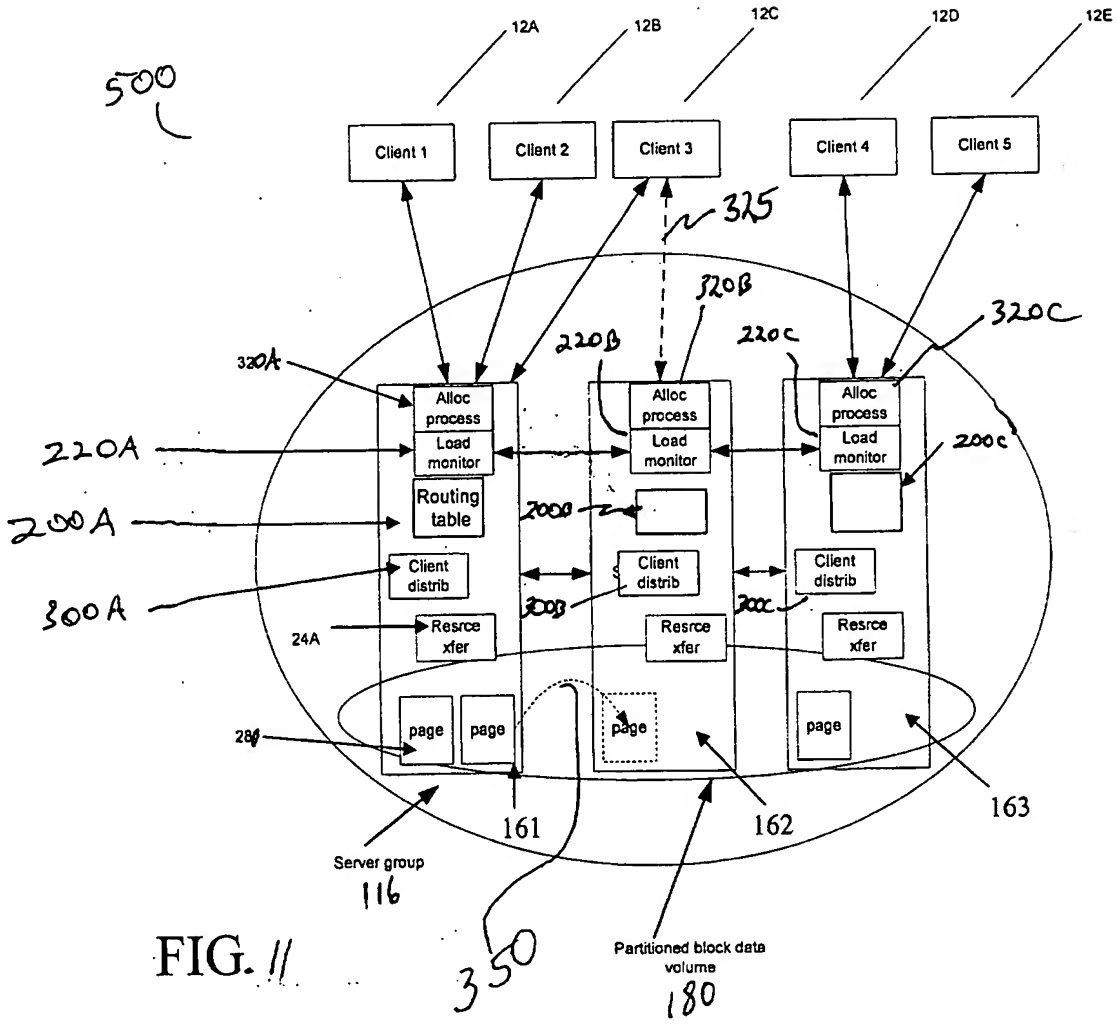

500
12A
12B
12C
12D
12E
Client 1
Client 2
Client 3
Client 4
Client 5
325
320A
320B
320C
220A
220B
220C
200A
200B
200C
300A
300B
300C
Alloc process
Load monitor
Routing table
Client distrib
Resrce xfer
24A
28A
page
161
162
163
Server group
116
350
Partitioned block data volume
180

FIG. 11

TECHNICAL APPENDIX
EQLC-P60-003

TITLE OF INVENTION:

EQLC-PXX-007

Distributed snapshot.

TECHNOLOGY TO PROTECT:

Every patent application is required to end with at least one concise statement, called a claim, of the subject matter that the patent is to protect. To this end, please provide a concise statement of the technology (system or process) you want this patent to protect. Focus on what you want to get a patent on for the company, and do not be concerned that you are not sure it is patentable. We will check the prior art to determine patentability before preparing the application. We will use your statement regarding what you want to protect to determine whether there is an available patent position for your company.

In a partitioned storage service, provided transparently by a group of servers, the storage "volumes" (analogous to disk drives in a conventional storage system) may be spread over several servers, with each server owning a portion of the data within the volume. Most operations are then performed simply by routing client requests to the owner of the particular piece of data operated on. (See invention disclosure "request routing for a partitioned application service" and "adaptive storage block data distribution".)

However, some operations act on the storage volume as a whole, rather than on individual blocks of data. In particular, this is true for the "snapshot" function.

"Snapshot" (also known by various other names, such as "point in time copy") is a service provided by many storage devices, in which a storage administrator can instruct the server to keep a copy of the state of the storage as of the time when the command is given. When this is done, the effect is that a second storage volume appears, which freezes the state of the original "main" volume. The main volume continues to be available and make be changes as needed after the snapshot is taken. Various techniques, such as "copy on write" are known to make this process efficient both as to time required to perform the action, and storage space required to record the "frozen view".

It is common practice for administrators to create snapshots in a controlled sequence of steps, where first the storage clients are instructed to suspend their use of the storage temporarily; then the snapshot is taken; finally the clients are instructed to continue normal operation. However, in a convention storage server this is often not necessary, because the snapshot is executed at some point in the sequence of operations on that server, so all operations preceding the snapshot command are reflected in the snapshot and none of the operations after the snapshot command are reflected in the snapshot.

For a partitioned service, this simplification does not apply because clients may be sending requests to any of the servers that comprise the partitioned service group. Those requests are then forwarded to the server that actually owns the data on which the requests act. Since this forwarding operation takes a variable amount of time, it would be possible for two commands to be acted on in an order different from the order in which the commands were sent by the storage clients. If a snapshot is created at that time, it may reflect a command issued by some client, but not reflect another command from some other client that was issued slightly earlier. The aim of this invention is to avoid such unexpected outcome, even when a snapshot command is issued while many client are actively issuing requests.

The solution involves imposing an ordering on the client requests and the snapshot command, such that the snapshot reflects all request made prior to a certain time and none of the requests made after that time. Several schemes are possible.

One approach uses a multi-stage operation. In the first stage, each server is notified that a snapshot has been requested. The servers then suspend processing of client requests. After all servers have done this, they are notified to take the snapshot. Finally, they all resume processing client requests.

A second approach uses time stamps, where client requests are marked with the time at which they are initially received by the server group. The snapshot command is likewise time-stamped. The snapshot is then constructed so it includes all requests with timestamps earlier than the snapshot command timestamp, and none of the requests with timestamps later than that.

The technology to be protected is these mechanisms for obtaining a consistent snapshot distributed among multiple servers. Minimally the first approach; preferably both of them.

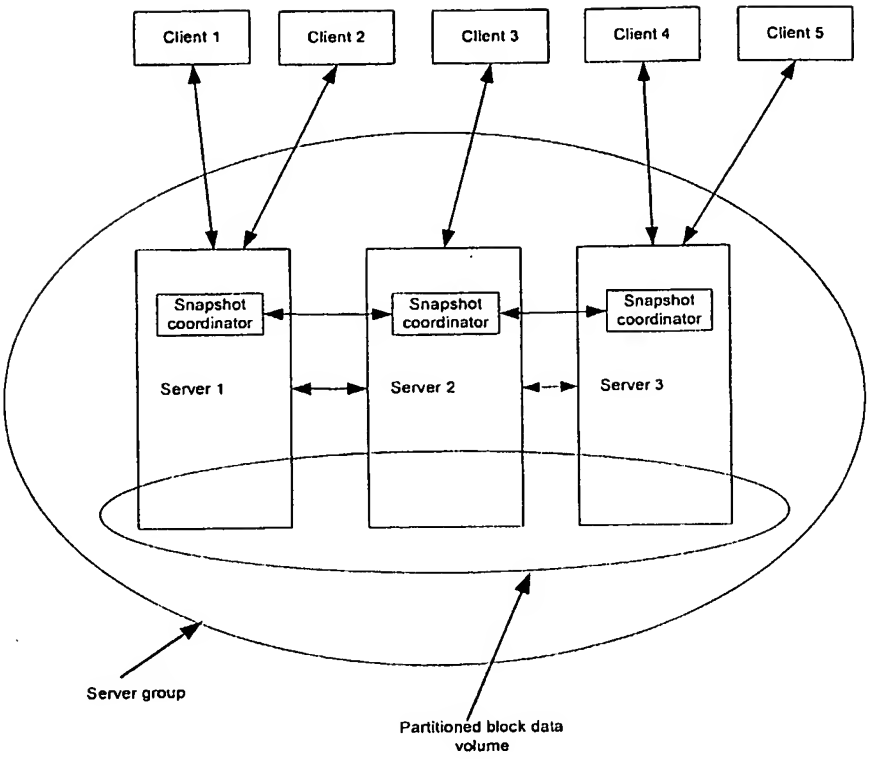
[redacted]

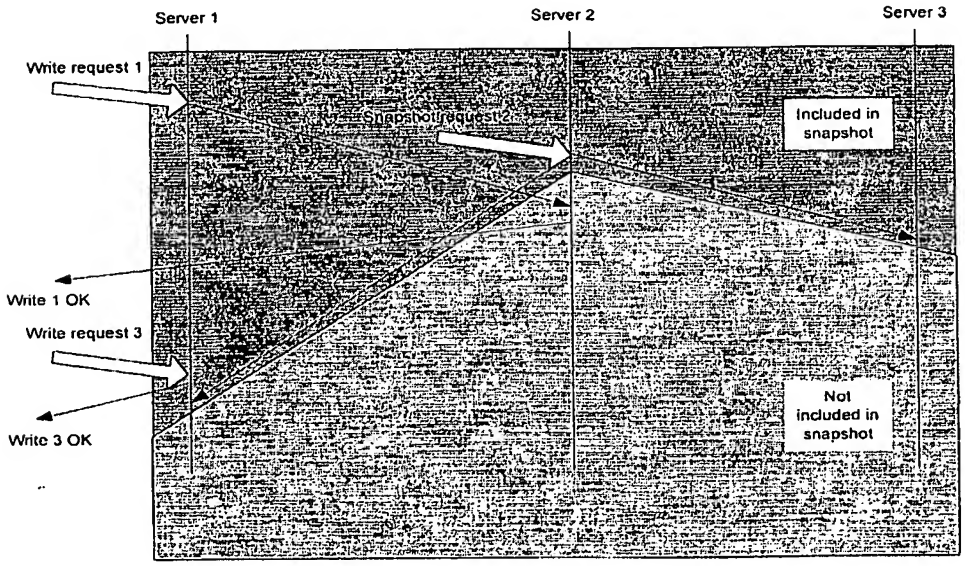
DESCRIPTION OF INVENTION:

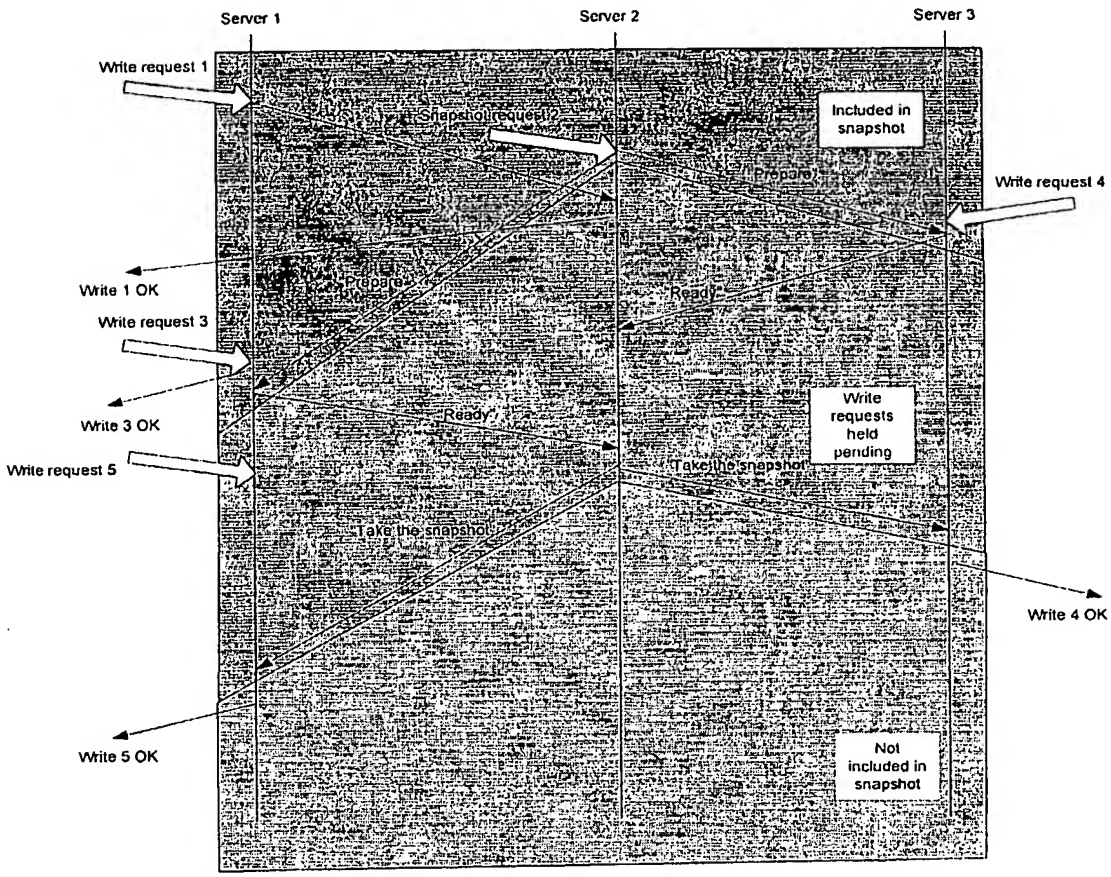
Please provide a brief description of your invention, making sure that your description is directed to the technology you want to protect as described in the section above. You can use any format you chose, but experience shows that it is most helpful to provide two or three simple drawings that illustrate the invention. These can be flow charts, functional block diagrams or suitable sketches. Examples of such drawings are provided at the end of this document. We prefer to receive the drawings in electronic form, such as PowerPoint, Visio or SmartDraw. Once the drawings are complete, please add one or two paragraphs that describe what is being shown in the drawings. Please make sure that you identify each element or step shown in the drawings. Consider including a summary of results achieved, features believed to be novel, further experimental work planned, and any additional information.

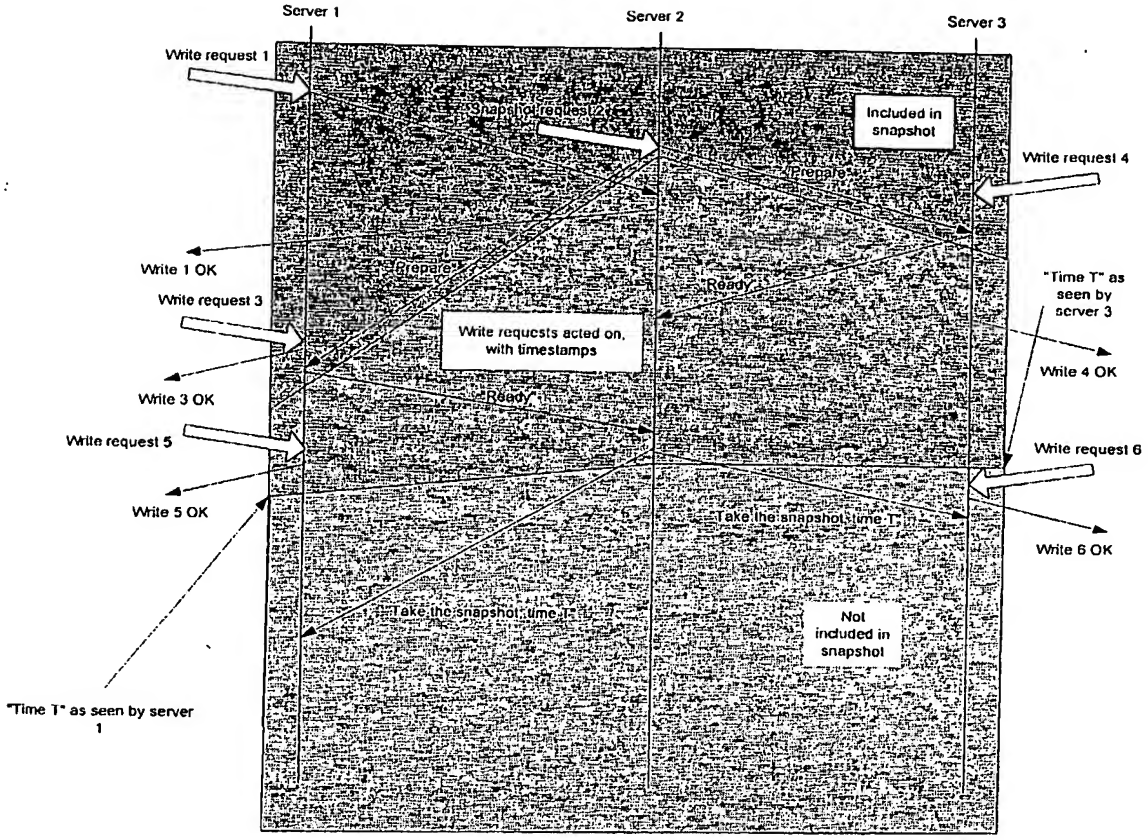
Refer to drawings in "distributed snapshot.vsd".
The first diagram shows a server group that provides a partitioned storage service. In this example, there are three servers, and a single volume whose data is partitioned among those three servers. Also, by way of illustration, 5 clients are shown.
The second diagram is a time/space diagram that shows how a sequence of operations including a snapshot request may have unexpected results if the invention is not used. In the diagram, the vertical lines represent three servers in the group, and the slanting lines represent requests (thick arrows) and responses (thin arrows). On the left, a client connected to server 1 issues two write requests. The first is forwarded to server 2, is processed, and acknowledged to indicate the request is complete. Then the second write ("write 3") is processed locally on server 1 and acknowledged. In the meantime, an administrator has sent a command to server 2 to create a snapshot. That command is forwarded to the other two servers for action. Each server takes action on the command when it is received. In the example shown, there is some delay in forwarding the snapshot command to server 1. The result in the example shown is that the second write ("write 3") is included in the snapshot, because it is seen at server 1 before the snapshot command, while the earlier write command ("write 1") is omitted from the snapshot, because it is seen at server 2 after the snapshot command. By normal storage service rules, this outcome is not acceptable – and even if acceptable it certainly is unexpected. The aim of the invention is to prevent this.
The third diagram ("invention case 1") shows the creation of a consistent distributed snapshot by a multi-stage sequence. The stages are coordinated by the server that processes the snapshot command from the administrator (server 2 in the example shown). The first action is to tell all servers to "prepare" for the snapshot, which means that any requests from storage clients that have not yet been answered will be held pending. All servers reply that they have done this ("Ready"). The coordinating server then commands "take the snapshot". All servers then create the snapshot, and having done so, release any requests that were held pending. In the example, writes 4 and 5 are held pending. The snapshot will contain all the requests completed in the green region (writes 1 and 3, in the example), and none that were requested in the yellow region. The ones in the blue region may or may not be in the snapshot; since they overlap the snapshot operation, that is an acceptable outcome by normal storage service rules.
The fourth diagram ("invention case 2") shows the creation of a consistent distributed snapshot through the use of timestamps. As before, the coordinating server (server 2 in this case) commands all other servers to prepare for a snapshot. In this case, requests that are still pending at this point are allowed to proceed and can be acknowledged as soon as they finish – it is not necessary to hold them pending. Instead, the servers note the time at which each such request was processed. In the example, this timestamping is done to writes 4,5, and 6. Once the coordinating server hears "ready" from each of the servers, it commands "take the snapshot" and includes in that command the current time. When the other servers see this command, they construct the snapshot such that write requests with timestamps earlier than T are included in the snapshot, and write requests with later timestamps are not. In the example, writes 4 and 5 are included in the snapshot, while write 6 is not.

[redacted]


Client 1
Client 2
Client 3
Client 4
Client 5
Snapshot coordinator
Snapshot coordinator
Snapshot coordinator
Server 1
Server 2
Server 3
Server group
Partitioned block data volume


Server 1
Server 2
Server 3
Write request 1
Snapshot request 2
Included in snapshot
Write 1 OK
Write request 3
Write 3 OK
Not included in snapshot


Server 1
Server 2
Server 3
Write request 1
Snapshot request?
Included in snapshot
Prepare?
Write request 4
Write 1 OK
Prepare?
Ready
Write request 3
Write 3 OK
Ready
Write requests held pending
Write request 5
Take the snapshot
Take the snapshot
Write 4 OK
Write 5 OK
Not included in snapshot


Server 1
Server 2
Server 3
Write request 1
Snapshot request 2
Included in snapshot
Write request 4
Prepare
Write 1 OK
Prepare
Ready
"Time T" as seen by server 3
Write request 3
Write requests acted on, with timestamps
Write 4 OK
Write 3 OK
Ready
Write request 5
Write request 6
Write 5 OK
Take the snapshot: time T
Write 6 OK
Take the snapshot: time T
Not included in snapshot
"Time T" as seen by server 1

[redacted] 6/26

TITLE OF INVENTION:

EQLC-PXX-006

Client load distribution.

TECHNOLOGY TO PROTECT:

Every patent application is required to end with at least one concise statement, called a claim, of the subject matter that the patent is to protect. To this end, please provide a concise statement of the technology (system or process) you want this patent to protect. Focus on what you want to get a patent on for the company, and do not be concerned that you are not sure it is patentable. We will check the prior art to determine patentability before preparing the application. We will use your statement regarding what you want to protect to determine whether there is an available patent position for your company.

In a partitioned application service, provided transparently by a group of servers, clients may connect to any server in the group, and request service. The servers forward client requests as needed, depending on the specific resource requested. For example, in a block storage service, requests are routed depending on which disk block number is requested. Since the service provided is the same no matter what server a client uses, client load balancing can be done to increase the overall capacity of the system proportionally to the number of servers that provide the partitioned service. The load balancing can be made adaptive to optimize the service provided to the clients as load changes.

The technology to be protected is – minimally -- the use of client load balancing for a partitioned storage service – where client data is partitioned over a group of servers. In that case, client load balancing may be done based on such parameters as the number of clients currently connected to each of the servers, the number of requests per second, the network load in packets or bytes per second, etc. If possible, more general claims (not limited to storage servers) would be desirable.

Load balancing has been done in the past in such settings as web servers. Unlike partitioned services of the kind we are building, load balancing in web servers normally involves a collection of servers all of which contain copies of the data to be served (i.e., the service is replicated rather than partitioned). Also, in web load balancing, the load balancing function is normally provided by a separate front-end device (a load balancing switch) rather than being integrated in the servers themselves.

DESCRIPTION OF INVENTION:

Please provide a brief description of your invention, making sure that your description is directed to the technology you want to protect as described in the section above. You can use any format you chose, but experience shows that it is most helpful to provide two or three simple drawings that illustrate the invention. These can be flow charts, functional block diagrams or suitable sketches. Examples of such drawings are provided at the end of this document. We prefer to receive the drawings in electronic form, such as PowerPoint, Visio or SmartDraw. Once the drawings are complete, please add one or two paragraphs that describe what is being shown in the drawings. Please make sure that you identify each element or step shown in the drawings. Consider including a summary of results achieved, features believed to be novel, further experimental work planned, and any additional information.

Refer to drawings in "Client load balancing.vsd".

The diagram shows a server group that provides a partitioned service; in this case, it is a block storage service. In this example, there are three servers, and a single volume whose data is partitioned among those three servers. Also, by way of illustration, 5 clients are shown.

In order to implement client load balancing, the servers implement an additional component: the load monitor. The load monitors observe those aspects of the system load that are of interest to the adaptive client load balancing algorithms. For example, they may track the number of clients connected to the server, the I/O request rates, the network load in packets per second and bytes per second, etc.

Client load balancing may occur at two times: (1) when a client initially makes contact with the server group, (2) subsequently, when a client has an active connection to the server group. For both cases, the server load monitor components track the current load at each server, and determine based on that information which server or servers

[redacted]

are best suited to accept more clients.
For case (1), a client initially contacts a server in the server group. It is often possible to spread the client load over several servers, for example using the "Round robin" feature of the standard Internet DNS (Domain Name Service) but this is often not effective due to client limitations. In any case, such spreading is not responsive to load. A more effective solution requires the servers to take explicit action to cause clients to contact those servers that are at any time the best suited to accept more clients. For example, in the block storage protocol iSCSI, a "redirect" message is available for this purpose.
For case (2), it may happen at times that client load becomes quite uneven but no new client connections occur. In that case, it is desirable to move existing connections from highly loaded servers to lightly loaded ones. As before, the load observations of the load monitors in each server drive this decision. The specific mechanism used to move an existing client depends on the protocol and service used. For example, with the block storage protocol iSCSI, the server can request the client to disconnect and re-establish its connection, at which time the server group reassigns the client according to case (1) above.
The diagram shows communication among the load monitors. The load monitors communicate in order to develop a group-wide analysis of the load. For example, the fact that a particular server has high I/O rates does not necessarily justify reducing the number of clients connected to it, but the fact that another server has a substantially lower load by comparison does.

The systems and methods described herein, include systems for managing requests for a plurality of clients for access to a set of resources. In one embodiment, the systems comprise a plurality of servers wherein the set of resources is partitioned across this plurality of servers. Each server has a load monitor process that is capable of communicating with the other load monitor processes for generating a measure of the client load on the server system and the client load on each of the respective servers.

Accordingly, in one embodiment, the systems comprise a server system having a plurality of servers, each of which has a load monitor process that is capable of coordinating with other load monitor processes executing on other servers to generate a system-wide view of the client load being handled by the server system and by individual respective servers.

Optionally, the systems may further comprise a client distribution process that is responsive to the measured system load and is capable of repartitioning the set of resources to thereby redistribute the client load.

Accordingly, it will be understood that the systems and methods described herein include client distribution systems that may work with a partitioned service, wherein the partitioned service is supported by a plurality of equivalent servers each of which is responsible for a portion of the service that has been partitioned across the equivalent servers. In one embodiment each equivalent server is capable of monitoring the relative load that each of the clients is placing on the system and on a particular respective server. Accordingly, each equivalent server is capable of determining when a particular client presents a relative burden. In one embodiment, the systems and methods described herein redistribute client load by, at least in part, redistributing resources across the plurality of servers.

In another embodiment, the systems and methods described herein include storage area network systems that may be employed for providing storage resources for an enterprise. The storage area network (SAN) of the invention comprises a plurality of servers and/or network devices. At least a portion of the servers and network devices operating on the storage area network include a load monitor process that monitors the client load being placed on the respective server or network device. The load monitor process is further capable of communicating with other load monitor processes operating on the storage area network. The load monitor process on the server is capable of generating a system-wide load analysis that indicate the client load being placed on the storage area network. Additionally, the load monitor process is capable of generating an analysis of the client load being placed on that respective server and/or network device. Based on the client load information observed by the load monitor process, the storage area network is capable of redistributing client load to achieve greater responsiveness to client requests. To this end, in one embodiment, the storage area network is capable of repartitioning the service supported by the system for the purpose of redistributing client load across the storage area network.

Further features and advantages of the invention will be apparent from the following description of preferred embodiments and from the claims.

Brief Description of the Drawings

The following figures depict certain illustrative embodiments of the invention in which like reference numerals refer to like elements. These depicted embodiments are to be understood as illustrative of the invention and not as limiting in any way.

FIG. 1 is a schematic diagram of a client-server architecture with servers organized in server groups;

FIG. 2 is a schematic diagram of the server groups as seen by a client;

FIG. 3 shows details of the information flow between the client and the servers of a group;

FIG. 4 is a process flow diagram for retrieving resources in a partitioned resource environment;

FIG. 5 depicts in more detail and as a functional block diagram one embodiment of a system according to the invention; and

FIG. 6 depicts an example of a routing table suitable for use with the system of FIG. 1.

Detailed Description of Certain Illustrated Embodiments

The systems and methods described herein include systems for organizing and managing resources that have been distributed over a plurality of servers on a data network. More particularly, the systems and methods described herein include systems and methods for providing more efficient operation of a partitioned service. The type of service can vary, however for purpose of illustration the invention will be described with reference to systems and methods for managing the allocation of data blocks across a partitioned volume of storage. It will be understood by those of skill in the art that the other applications and services may include, although are not limited to, distributed file systems, systems for supporting application service providers and other applications. Moreover, it will be understood by those of ordinary skill in the art that the systems and methods described herein are merely exemplary of the kinds of systems and methods that may be achieved through the invention and that these exemplary embodiments may be modified, supplemented and amended as appropriate for the application at hand.

Referring first to FIG. 1 one embodiment of a system according to the invention is depicted. As show in FIG. 1, one or several clients 12 are connected, for example via a network 14, such as the Internet, an intranet, a WAN or LAN, or by direct connection, to servers 161, 162, and 163 that are part of a server group 16.

The depicted clients 12 can be any suitable computer system such as a PC workstation, a handheld computing device, a wireless communication device, or any other such device, equipped with a network client program capable of accessing and interacting with the server group 16 to exchange information with the server group 16. Optionally, the client 12 and the server group 16 rely on an unsecured communication path for accessing services at the remote server group 16. To add security to such a communication path, the client 12 and the server group 16 may employ a security system, such as any of the conventional security systems that have been developed to provide a secure channel for transmitting data over a network. One such system is the Netscape secured socket layer (SSL) security mechanism that provides a trusted path between a client and a server.

FIG. 1 further depicts that the client 12 may communicate with a plurality of servers, 161, 162 and 163. The servers 161, 162 and 163 employed by the system 10 may be conventional, commercially available server hardware platforms, such as a Sun Sparc™ systems running a version of the Unix operating system. However any suitable data processing platform may be employed. Moreover, it will be understood that one or more of the servers 161, 162 or 163 may comprise a network device, such as a tape library, or other device, that is networked with the other servers and clients through network 14.

Each server 161, 162 and 163 may include software components for carrying out the operation and the transactions described herein, and the software architecture of the servers 161, 162 and 163 may vary according to the application. In certain embodiments, the servers 161, 162 and 163 may employ a software architecture that builds certain of the processes described below into the server's operating system, into device drivers, into application level programs, or into a software process that operates on a peripheral device, such as a tape library, a RAID storage system or some other device. In any case, it will be understood by those of ordinary skill in the art, that the systems and methods described herein may be realized through many different embodiments, and practices, and that the particular embodiment and practice employed will vary as a function of the application of interest and all these embodiments and practices fall within the scope hereof.

In operation, the clients 12 will have need of the resources partitioned across the server group 16. Accordingly, each of the clients 12 will send requests to the server group 16. The clients 12 typically act independently, and as such, the client load placed on the server group 16 will vary over time. In a typical operation, a client 12 will contact

one of the servers, for example server 161, in the group 16 to access a resource, such as a data block, page, file, database table, application, or other resource. The contacted server 161 itself may not hold or have control over the requested resource. However, in a preferred embodiment, the server group 16 is configured to make all the partitioned resources available to the client 12 regardless of the server that initially receives the request. For illustration, the diagram shows two resources, one resource 18 that is partitioned over all three servers, servers 161, 162, 163, and another resource 17 that is partitioned over two of the three servers. In the exemplary application of the system 10 being a block data storage system, each resource 18 and 17 may represent a partitioned block data volume.

The depicted server group 16 therefore provides a block data storage service that may operate as a storage area network (SAN) comprised of a plurality of equivalent servers, servers 161, 162 and 163. Each of the servers 161, 162 and 163 may support one or more portions of the partitioned block data volumes 18 and 17. In the depicted system 10, there are two data volumes and three servers, however there is no specific limit on the number of servers. Similarly, there is no specific limit on the number of resources or data volumes. Moreover, each data volume may be contained entirely on a single server, or it may be partitioned over several servers, either all of the servers in the server group, or a subset of the server group. In practice, there may of course be limits due to implementation considerations, for example the amount of memory available in the servers 161, 162 and 163 or the computational limitations of the servers 161, 162 and 163. Moreover, the grouping itself, i.e., deciding which servers will comprise a group, may in one practice involve an administrative decision. In a typical scenario, a group might at first contain only a few servers, perhaps only one. The system administrator would add servers to a group as needed to obtain the level of performance required. Increasing servers creates more space (memory, disk storage) for resources that are stored, more CPU processing capacity to act on the client requests, and more network capacity (network interfaces) to carry the requests and responses from and to the clients. It will be appreciated by those of skill in the art that the systems described herein are readily scaled to address increased client demands by adding additional servers into the group 16. However, as client load varies, the system 10 as described below can redistribute client load to take better advantage of the available resources in server group 16. To this end, the system 10 in one embodiment, comprises a plurality of equivalent servers. Each equivalent server supports a portion of the resources partitioned over the server group 16. As client requests are delivered to the equivalent servers, the equivalent servers coordinate among themselves to generate a measure of system load and to generate a measure of the client load of each of the equivalent servers. In a preferred practice, this coordinating is done in a manner that is transparent to the clients 12, so that the clients 12.

Referring now to FIG. 2, a client 12 connecting to a server 161 (FIG. 1) will see the server group 16 as if the group were a single server having multiple IP addresses. The client 12 is not aware that the server group 16 is constructed out of a potentially large number of servers 161, 162, 163, nor is it aware of the partitioning of the block data volumes 17, 18 over the several servers 161, 162, 163. As a result, the number of servers and the manner in which resources are partitioned among the servers may be changed without affecting the network environment seen by the client 12.

FIG. 3 shows the resource 18 of FIG. 2 as being partitioned across servers 161, 162 and 163. In the partitioned server group 16, any data volume may be spread over any number of servers within the group 16. As seen in FIGS. 1 and 2, one volume 17 (Resource 1) may be spread over servers 162, 163, whereas another volume 18 (Resource 2) may be spread over servers 161, 162, 163. Advantageously, the respective volumes may be arranged in fixed-size groups of blocks, also referred to as "pages", wherein an exemplary page contains 8192 blocks. Other suitable page sizes may be employed. In an exemplary embodiment, each server in the group 16 contains a routing table 165 for each volume, with the routing table 165 identifying the server on which a specific page of a specific volume can be found. For example, when the server 161 receives a request from a client 12 for volume 3, block 93847, the server 161 calculates the page number (page 11 in this example for the page size of 8192) and looks up in the routing table 165 the location or number of the server that contains page 11. If server 163 contains page 11, the request is forwarded to server 163, which reads the data and returns the data to the server 161. Server 161 then send the requested data to the client 12. The response may be returned to the client 12 via the same server 161 that received the request from the client 12.

Accordingly, it is immaterial to the client 12 as to which server 161, 162, 163 has the resource of interest to the client 12. As described above, the servers 162, 162 and 163 will employ the routing tables to service the client request, and the client 12 need not know ahead of time which server is associated with the requested resource. This allows portions of the resource to exist at different servers. It also allows resources, or portions thereof, to be moved while the client 12 is connected to the server group 16. Upon moving a resource, the routing tables 165 are updated

as necessary and subsequent client requests will be forwarded to the server now responsible for handling that request. At least within a resource 17 or 18, the routing tables 165 may be identical.

FIG. 4 depicts an exemplary client allocation process 40 for handling client requests in a partitioned server environment. The client allocation process 40 begins at 41 by receiving a request for a resource, such as a file or blocks of a file, at 42. The client allocation process 40 checks, in operation 43, if the requested resource is present at the initial server that received the request from the client 12. If the requested resource is present at the initial server, the initial server returns the requested resource to the client 12, at 48, and the process 40 terminates at 49. Conversely, if the requested resource is not present at the initial server, the server will consult a routing table, operation 44, to determine which server actually holds the resource requested by the client, operation 45. The request is then forwarded to the server that holds the requested resource, operation 46, which returns the requested resource to the initial server, operation 48. The process 40 then goes to 48 as before, to have the initial server forward the requested resource to the client 12, and the process 40 terminates, at 49.

Accordingly, one of ordinary skill in the act will see that the system and methods described herein are capable of partitioning one or more resources over a plurality of servers thereby providing a server group capable of handling requests from multiple clients. Additionally, the above description illustrates that the systems and methods described herein can redistribute or repartition the resource to change how portions of the resource are distributed or spread across the server group. The resources spread over the several servers can be directories, individual files within a directory, blocks within a file or any combination thereof. Other partitioned services may be realized. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned server that supports applications being delivered over the Internet. In general, the approach can be applied to any service where a client request can be interpreted as a request for a piece of the total resource.

Turning now to FIG. 5, one particular embodiment of the system 10 is depicted wherein the system is capable of redistributing client load to provide more efficient service. Specifically, FIG. 5 depicts the system 10 wherein the clients 12 communicate with the server block 16. The server block 16 includes three servers, server 161, 162 and 163. In the embodiment of FIG. 5 the servers 161, 162 and 163 are equivalent servers, in that each of the servers will provide substantially the same resource to the same request from a client. As such, from the perspective of the clients 12, the server group 16 appears to be a single server system that provides multiple network or IP addresses for communicating with clients 12. Each server includes a routing table, depicted as routing tables 20A, 20B and 20C, a load monitor process 22A, 22B and 22C respectively, a client allocation process 32A, 32B, and 32C, a client distribution process 30A, 30B, 30B and 30C and a resource transfer process, 24A, 24B and 24C respectively. Further, and for the purpose of illustration only, the FIG: 5 represents the resources as pages of data 28 that may be transferred from one server to another server.

As shown in FIG. 5, each of the routing tables 20A, 20B and 20C are capable of communicating with each other for the purpose of sharing information. As described above, the routing tables can track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 16. In the embodiment shown in FIG. 5 the server group 16 may form a SAN wherein each of the equivalent servers 161, 162 and 163 has an individual IP address that may be employed by a client 12 for accessing that particular equivalent server on the SAN. As further described above, each of the equivalent servers 161, 162 and 163 may be capable of providing the same response to the same request from a client 12. To that end, the routing tables 20A, 20B and 20C of the individual equivalent 161, 162 and 163 coordinate with each other to provide a global database of the different resources, and the specific equivalent servers that are responsible for those resources.

FIG. 6 depicts one example of a routing table 20A and the information stored therein.

As depicted in FIG. 6, each routing table includes an identifier for each of the equivalent servers 161, 162 and 163 that support the partitioned data block storage service. Additionally, each of the routing tables includes a table that identifies those data blocks associated with each of the respective equivalent servers. In the embodiment depicted by FIG. 6, the equivalent servers support two partitioned volumes. A first one of the volumes is distributed or partitioned across all three equivalent servers 161, 162 and 163. The second partitioned volume is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

Turning back to FIG. 5, the system 10 illustrates and embodiment of the invention wherein the individual servers 161, 162 and 163 are capable of redistributing client load across the different servers 161, 162 and 163. In this way, the systems and methods of the invention are capable of dealing with the dynamic behavior of client load. Accordingly, from time to time it may be that one or more of the servers within the server group 16 are over-utilized, either in total or in relation to the other servers. Accordingly, the systems and methods described herein may

improve efficiencies and provide more prompt client responses by distributing client load across the different servers within the server group 16 and thereby more efficiently using the resources provided by the server group 16.

FIG. 5 depicts an example of such a client load redistribution. Specifically, FIG. 5 depicts that a plurality of clients 12 are capable of communicating with individual ones of the servers 161, 162 and 163. FIG. 5 illustrates pictorially that the system 10 may be capable of redistributing client load by having client 12C, that is originally communicating with server 161, redistributed to server 162. To this end, FIG. 5 depicts an initial condition wherein the server 161 is communicating with clients 12A, 12B and 12C. This is depicted in FIG. 5 by the bidirectional arrows coupling the server 161 to the respective clients 12A, 12B and 12C. As further shown in FIG. 5, upon an initial condition, clients 12D and 12E are communicating with server 163 and no client during the initial condition is communicating with server 162. Accordingly, during this initial condition, server 161 is supporting requests from three clients, clients 12A, 12B and 12C. Additionally, FIG. 5 shows that during the initial condition server 162 is not servicing or responding to requests from any of the clients.

Accordingly, in this initial condition the system 10 may determine that server 161 is overly burdened or resource constrained. This determination may result from an analysis that server 161 is now resource constrained in that it is overly utilized given the resources available. For example, it could be that the server 161 has limited memory and that the requests being generated by clients 12A, 12B and 12C have overburdened the memory resources available to server 161. Thus, server 161 may be responding to client requests at a level of performance that is below an acceptable limit. Alternatively, it may be determined that server 161, although performing and responding to client requests at an acceptable level, is overly burdened with respect to the client load being carried by server 162. Accordingly, the system 10 may make a determination that overall efficiency for the server group 16 may be improved by redistributing client load from its initial condition to one wherein server 162 services requests from client 12C. To this end, FIG. 5 depicts this redistribution of client load by illustrating a connection between client 12C and server 162 where this connection is depicted by a dotted bi-directional arrow connected between client 12C and server 162. It will be understood that after redistribution of the client load, the communication path between the client 12C and server 161 will terminate.

In operation, each of the depicted servers 161, 162 and 163 is capable of monitoring the complete load that is placed on the server group 16 as well as the load from each client and the individual client load that is being handled by each of the respective servers 161, 162 and 163. To this end, each of the servers 161, 162 and 163 include a load monitoring process 22A, 22B and 22C respectively. As described above, the load monitor processes 22A, 22B and 22C are capable of communicating among each other. This is illustrated in FIG. 5 by the bidirectional lines that couple the load monitor processes on the different servers 161, 162 and 163. Each of the depicted load monitor processes may be software processes executing on the respective servers and monitoring the client requests being handled by the respective server. The load monitors may monitor the number of individual clients 12 being handled by the respective server, the number of requests being handled by each and all of the clients 12, and other information.

Accordingly, the loan monitor process 22A is capable of generating information representative of the client load applied to the server 161 and is capable of corresponding with the load monitor 22B of server 162. In turn, the load monitor process 22B of server 162 may communicate with the load monitor process 22C of server 163. By allowing for communication between the different load monitor processes 20A, 20B and 20C, the load monitor processes may determine the system-wide load applied to the server group 16 by the clients 12.

Figure 5 further illustrates that server 161 as well as the other servers, includes a client distribution process 30A, 30B and 30C respectively. In one embodiment, the client distribution process 30A is capable of processing information generated by the load monitor process 22A and employing that information for the purpose of distributing client load among the servers of the server group 16. To this end, the client distribution process 30A may be software process executing on server 161 and capable of processing the system load and client load information generated by the load monitor process to determine whether server 161 is servicing a disproportionate share of the client requests being handled by server group 16. In one practice, the client distribution process 30A determines from the loan monitor process 22A whether the server 161 is servicing or responding to too many of the clients 12. Upon a determination that server 161 is overburdened, the client distribution process 30A may determine from the system load information whether there is another server in server group 16 that is more available to handle a portion of the clients 12 being serviced by server 161. In the depiction of this process illustrated by FIG. 5, the client distribution process 30A determines that server 162 would be more appropriate for handling at least one of the clients 12 being supported by server 161.

More particularly, FIG. 5 depicts that the client distribution process 30A determines that client 12C is to be redistributed from server 161 to server 162.

In the example depicted in FIG. 5, the client 12C may be continually requesting access to the same resource. In this case, FIG. 5 shows that the resource may be the page 28 maintained by the server 161. For the purpose of distributing client 12C to server 162, the client distribution process 30A can activate a resource transfer process 24A that transfers page 28 from server 161 to server 162. Accordingly, in the embodiment depicted in FIG. 5 the client distribution process 30A cooperates with the resource transfer process 24A to re-partition the resources in a manner that is more likely to cause client 3 to continually make requests to server 162 as opposed to server 161.

Once the resource 28 has been transferred to server 162, the routing table 20B can update itself and update the routing tables 20A and 20C respectively. In this way, the resources may be repartitioned across the servers 161, 162 and 163 in a manner that may redistribute client load as well.

In an alternate embodiment, the client distribution process 30A cooperates with the client allocation process 32A. As described above, the client allocation process 32A is capable of determining the proper server for servicing the client request. In one embodiment, the client distribution process 30A operates to redistribute the clients 12 being serviced by the client allocation process 32A. To this end, the client distribution process can communicate with the load monitor process 22A to determine a better distribution of client load for the server group 16. Upon determining the better distribution, the client distribution process 30A may communicate with the allocation process 32A to cause a selected one or ones of the clients 12 communicating with server 161 to now communicate with server 162. Once reallocated, server 162 may be better available for handling client requests from the clients 12 and allocating or otherwise determining which of the servers in server group 16 has the resource that is the subject matter of the request made by the client of interest.

In one embodiment, the client allocation process 32A responds to the client distribution process 30A to implement a "round-robin" process that sequentially distributes incoming client requests among the different servers of the server group 16. In other embodiments, the client allocation process is employed to reallocate clients 12 based on an adaptive process that seeks to achieve an optimal or substantially optimal distribution of client load among the servers and server group 16. In operation, the client allocation process may employ a redirection process that redirects incoming clients such that the incoming client is moved from one server to another server.

In a further alternative embodiment, the client allocation process 32A may employ a dynamic reallocation process that can disconnect a client 12 during a session with the respective server 161. The client allocation process 32A may also reallocate the appropriate resources from one server to another. Thus, as described above resources maintained on server 161 may be transferred to server 162. Once transferred, the client allocation process may cause the client 12C to request resources from server 162. In this way, if during the servicing of a plurality of clients one or more servers handling such client requests becomes overburdened, the overburdened server may discontinue its service of a particular client and that service may be transferred over to a different server that had more resources or as otherwise more available to service such requests.

Although FIG. 1 depicts the system as an assembly of functional block elements including a group of server systems, it will be apparent to one of ordinary skill in the art that the systems of the invention may be realized as computer programs or portions of computer programs that are capable of running on the servers to thereby configure the servers as systems according to the invention. Moreover, although FIG. 1 depicts the group 16 as a local collection of servers, it will be apparent to those or ordinary skill in the art that this is only one embodiment, and that the invention may comprise a collection or group of servers that includes server that are physically remote from each other.

As discussed above, in certain embodiments, the systems of the invention may be realized as software components operating on a conventional data processing system such as a Unix workstation. In such embodiments, the system can be implemented as a C language computer program, or a computer program written in any high level language including C++, Fortran, Java or basic. General techniques for such high level programming are known, and set forth in, for example, Stephen G. Kochan, *Programming in C*, Hayden Publishing (1983).

While the invention has been disclosed in connection with the preferred embodiments shown and described in detail, various modifications and improvements thereon will become readily apparent to those skilled in the art. Accordingly, the spirit and scope of the present invention is to be limited only by the following claims.

A system for managing requests from a plurality of clients for access to a set of resources, comprising

a plurality of servers having the set of resources partitioned thereon, and having

a load monitor process capable of communicating with other load monitor processes for generating a measure of system load, and a measure of client load on respective ones of the plurality of servers.

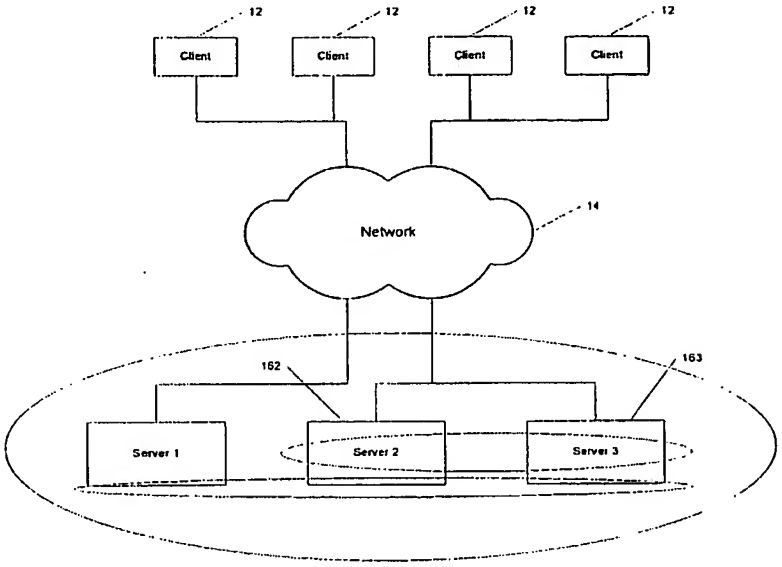


FIG. 1

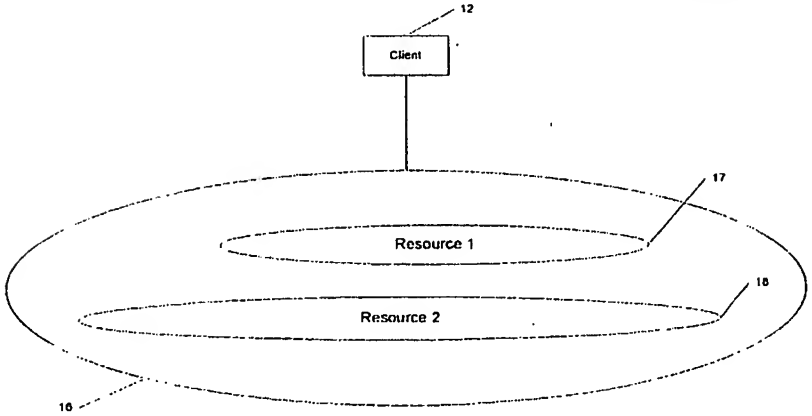


FIG. 2

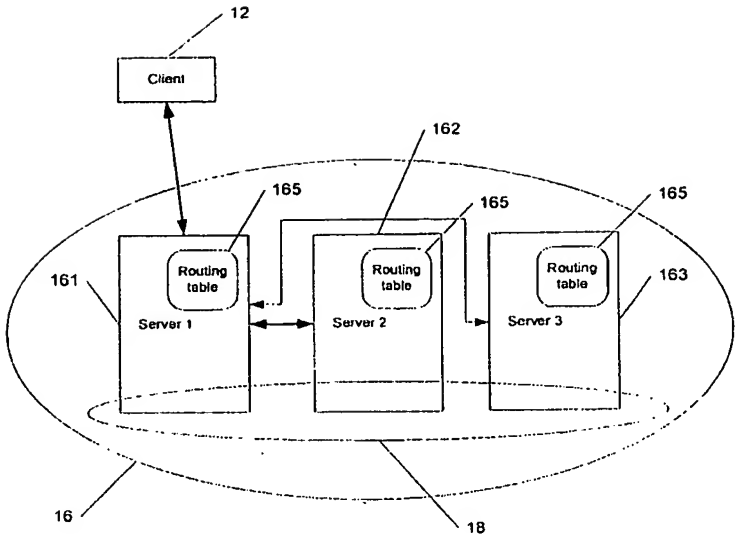

12
Client
162
165
165
165
Routing table
Routing table
Routing table
161
163
Server 1
Server 2
Server 3
16
18

FIG. 3

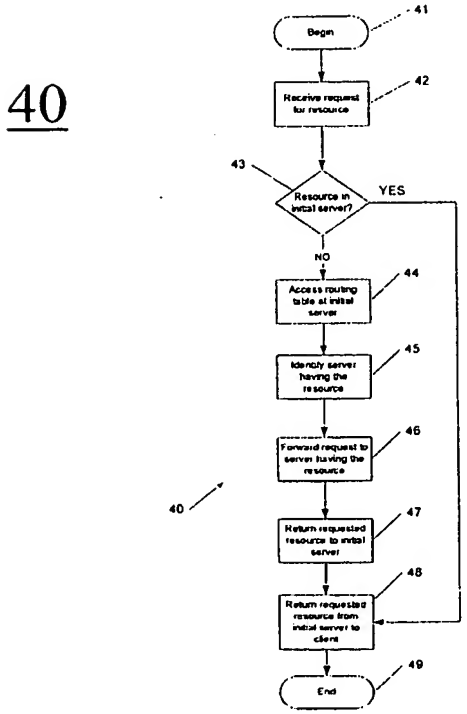

40
41
Begin
42
Receive request for resource
43
Resource in initial server?
YES
NO
44
Access routing table at initial server
45
Identify server having the resource
46
Forward request to server having the resource
47
Return requested resource to initial server
48
Return requested resource from initial server to client
49
End

FIG. 4

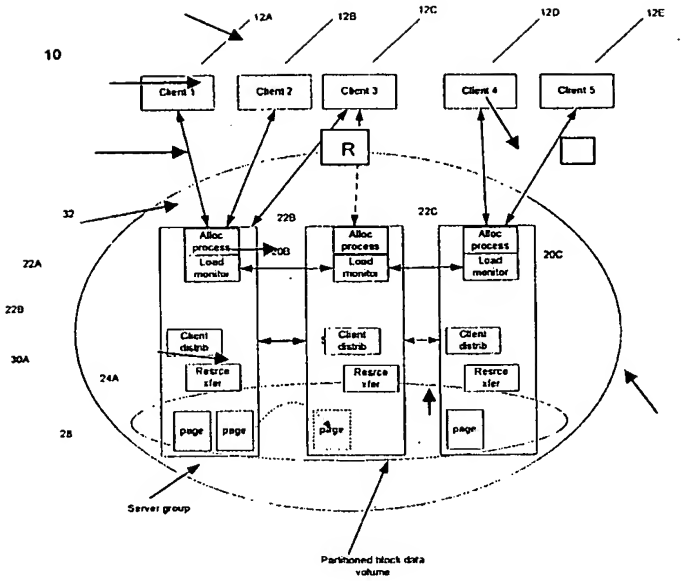

10
12A
12B
12C
12D
12E
Client 1
Client 2
Client 3
Client 4
Client 5
R
32
22A
22B
22C
20A
20B
20C
Alloc process
Load monitor
Client distrib
Resrce xfer
30A
24A
28
page
Server group
Partitioned block data volume

FIG 5

TITLE OF INVENTION: EQLC-PXX-005

Differentiated Storage Pools over a common set of storage devices with multiple RAID levels on each device.

TECHNOLOGY TO PROTECT:

Every patent application is required to end with at least one concise statement, called a claim, of the subject matter that the patent is to protect. To this end, please provide a concise statement of the technology (system or process) you want this patent to protect. Focus on what you want to get a patent on for the company, and do not be concerned that you are not sure it is patentable. We will check the prior art to determine patentability before preparing the application. We will use your statement regarding what you want to protect to determine whether there is an available patent position for your company.

In a block storage environment with multiple devices, differentiated pools of storage are created by applying multiple RAID techniques simultaneously on each device. These pools of storage differ in regards to performance and reliability. Adaptive load balancing can be used to place data in pools to optimize service for a single or multiple consumers of storage services.

For example:

a) Storage devices offer different performance across their LBN space: some LBN ranges have higher bandwidth for read and/or write, some LBN ranges have higher throughput for small and/or random operations, some LBN ranges have slower bandwidth, and some LBN ranges have slower throughput.
b) RAID levels have different performance characteristics for bandwidth and throughput for read and write operations, as well as in both normal mode and degraded operating modes.
c) Device performance differences across LBN space, and performance and reliability differences of RAID techniques, can be combined of to create storage pools that offer different classes of service across a common set of storage devices to the consumers of storage services.

Adaptive load balancing provides the ability to adjust data placement in pools for continuous optimal performance.

The technology to be protected is – minimally – :

1) The use of multiple raid levels concurrently on each device to create differentiated classes of storage service.
2) The use of multiple raid levels concurrently on each device in combination with device LBN performance differences to create differentiated classes of storage service.
3) Load balancing storage services across multiple differentiated classes of storage service:
    a. Automatic, adaptive load balancing of a single or multiple storage services (volumes or file systems) across pools via
        i. Volume based allocation
        ii. Extent based allocation
        iii. Page-based allocation
    b. Manual load balancing of whole volumes

DESCRIPTION OF INVENTION:

Please provide a brief description of your invention, making sure that your description is directed to the technology you want to protect as described in the section above. You can use any format you chose, but experience shows that it is most helpful to provide two or three simple drawings that illustrate the invention. These can be flow charts, functional block diagrams or suitable sketches. Examples of such drawings are provided at the end of this document. We prefer to receive the drawings in electronic form, such as PowerPoint, Visio or SmartDraw. Once the drawings are complete, please add one or two paragraphs that describe what is being shown in the drawings. Please make sure that you identify each element or step shown in the drawings. Consider including a summary of results achieved, features believed to be novel, further experimental work planned, and any additional information.

[redacted]

Refer to drawings in "Differentiated Storage Pools.vsd".

Diagram 1: an individual storage device that has different performance characteristics based on a list of LBNs

Diagram 2: RAID differentiated storage pools sharing the same storage devices.

Diagram 3: Differentiated storage pools by combining individual storage device performance and RAID performance characteristics.

Diagram 4: Extent based and page based allocation of a storage service across differentiated storage pools. Adaptive load balancing is used to initially place, and automatically update allocation based on space and performance utilization of the storage service.

[redacted]

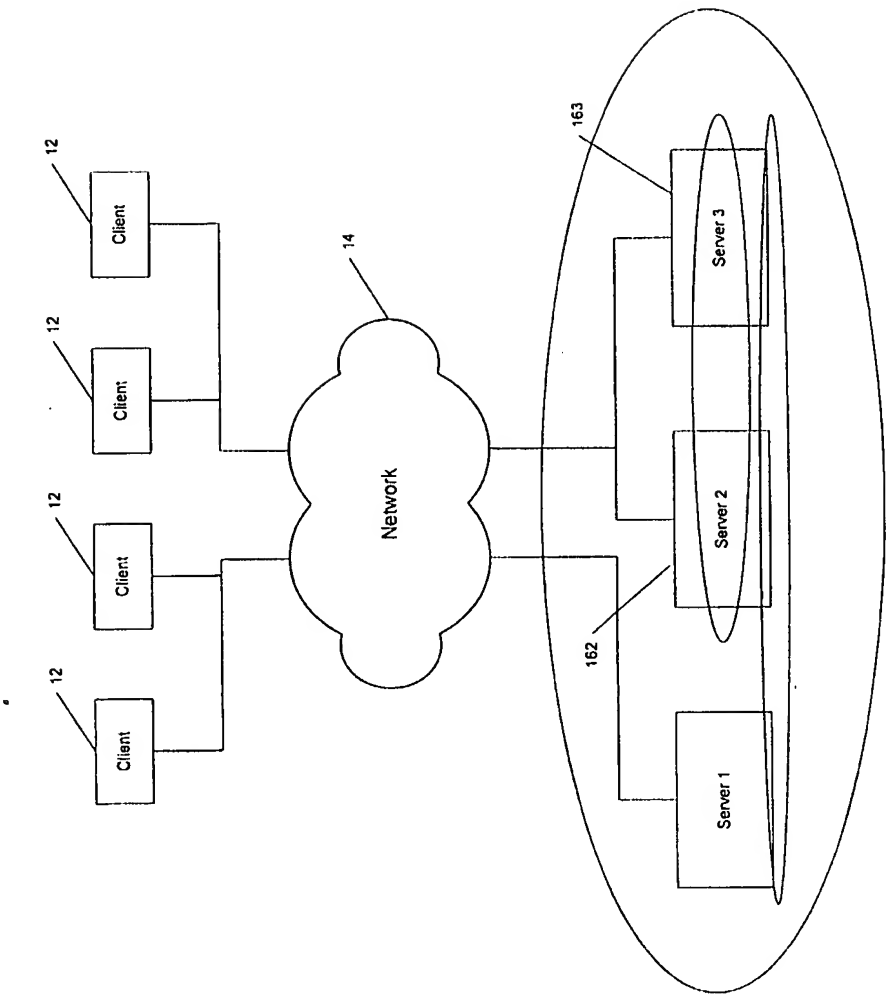

12
Client
12
Client
12
Client
12
Client
14
Network
163
Server 3
Server 2
162
Server 1

FIG. 1

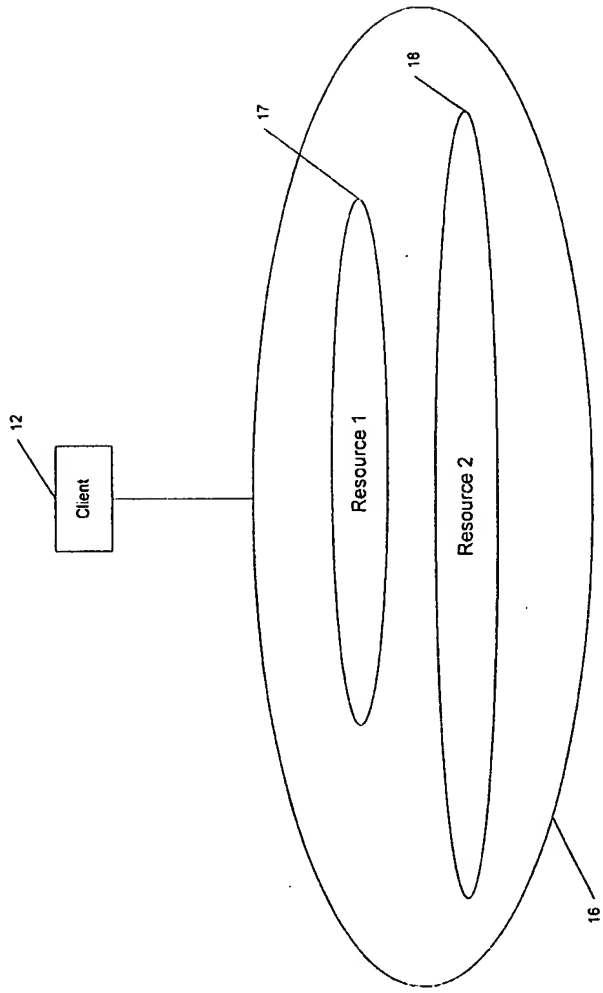


FIG. 2

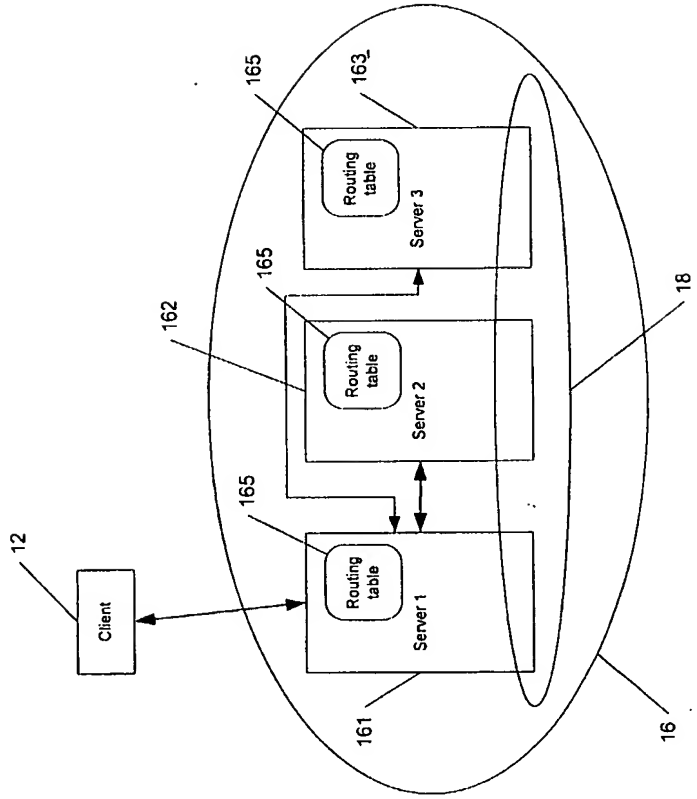

12
Client
161
Server 1
Routing table
165
162
Server 2
Routing table
165
163
Server 3
Routing table
165
16
18

FIG. 3

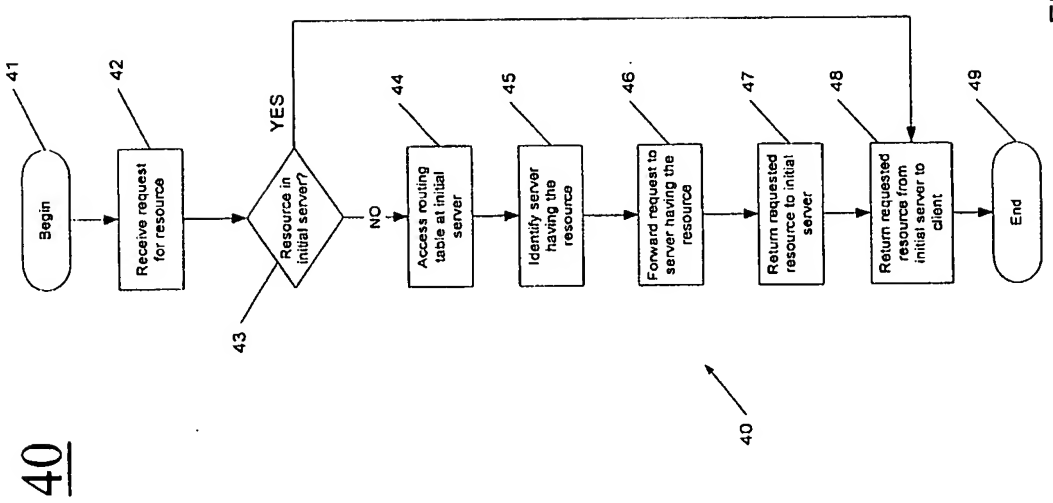

40
41
Begin
42
Receive request for resource
43
Resource in initial server?
YES
NO
44
Access routing table at initial server
45
Identify server having the resource
46
Forward request to server having the resource
47
Return requested resource to initial server
48
Return requested resource from initial server to client
49
End
40

FIG. 4

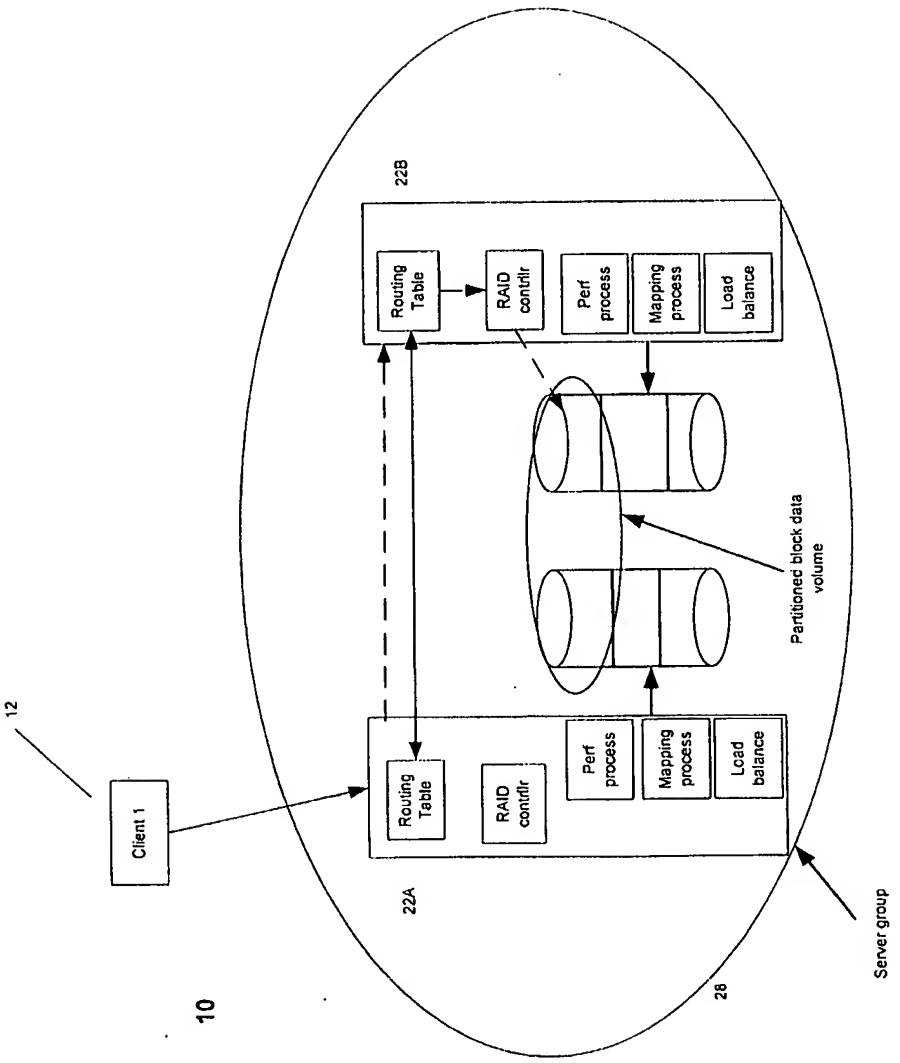

12
Client 1
10
22A
Routing Table
RAID contrlr
Perf process
Mapping process
Load balance
22B
Routing Table
RAID contrlr
Perf process
Mapping process
Load balance
Partitioned block data volume
28
Server group

FIG 5

Diagram 1
Storage device
performance differences
by LBN list

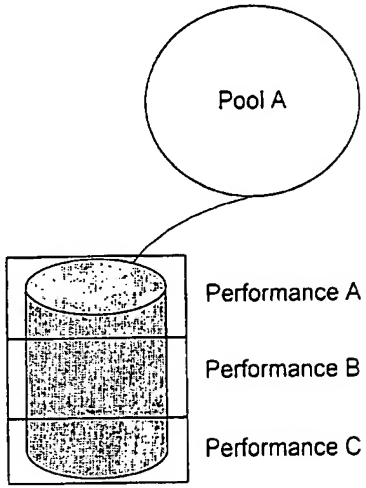


FIG. 6

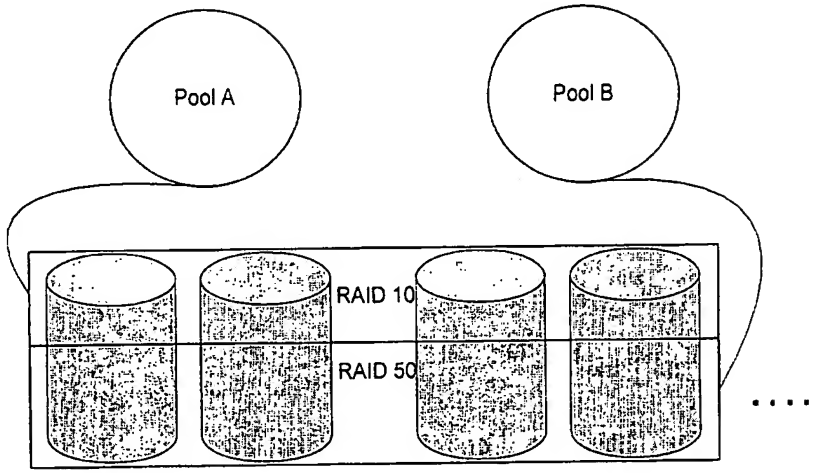


FIG. 7

Diagram 3
Differentiated Pools by combining RAID and device performance

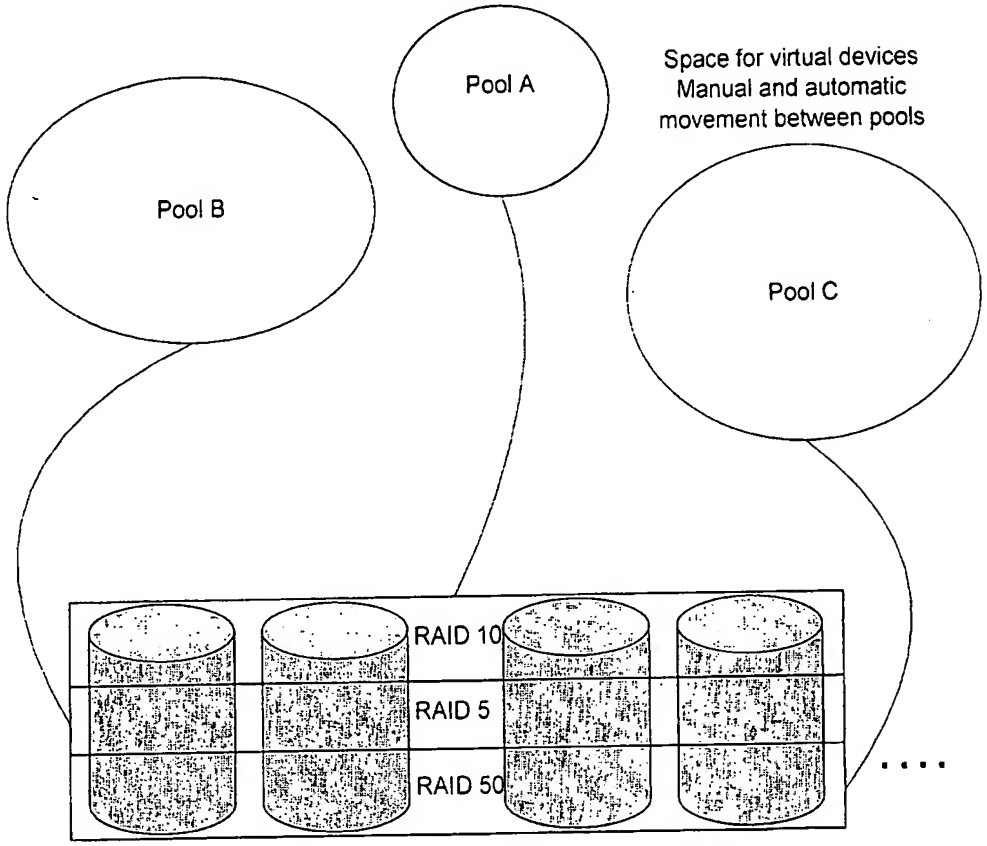


FIG. 8

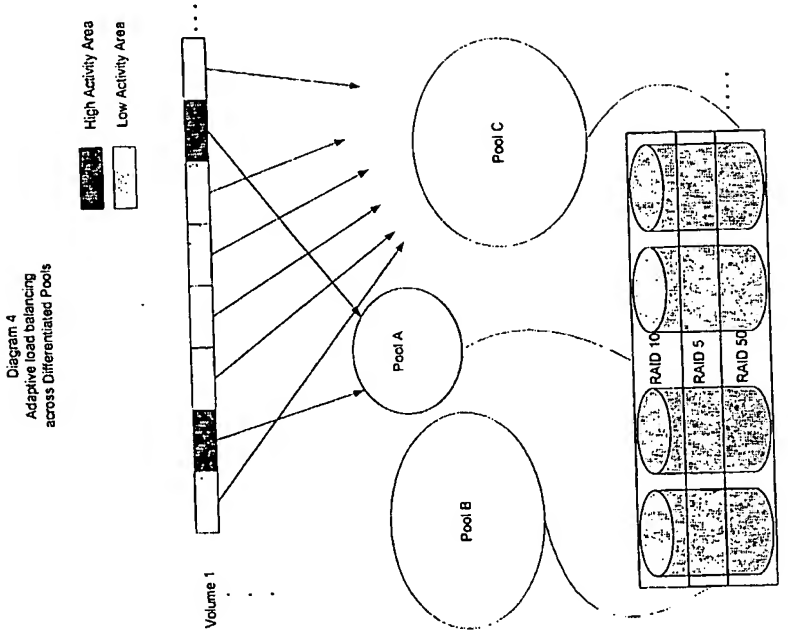


Diagram 4
Adaptive load balancing
across Differentiated Pools
High Activity Area
Low Activity Area
Volume 1
Pool A
Pool B
Pool C
RAID 10
RAID 5
RAID 50

FIG. 9

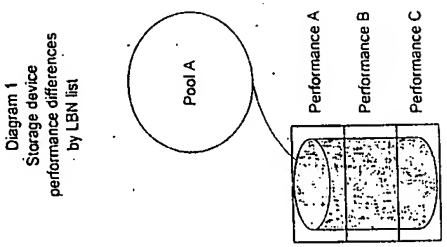


FIG. 6

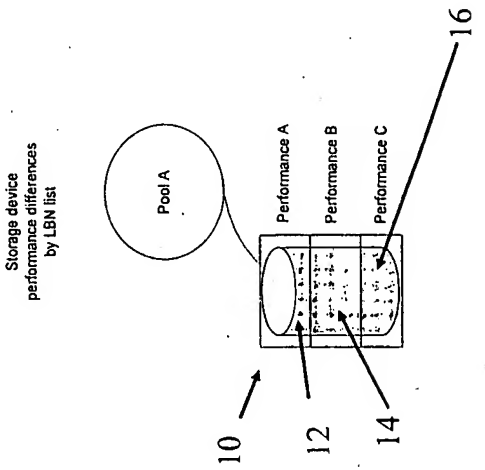


FIG. 1

From U.S. Provisional Application No. 60/441,810
Filed January 21, 2003

From U.S. Utility Application No. 10/761,884
Filed January 20, 2004

EXHIBIT

tabbies

B

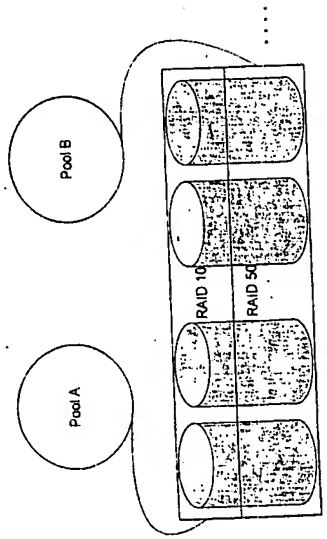


FIG. 7

From U.S. Provisional Application No. 60/441,810
Filed January 21, 2003

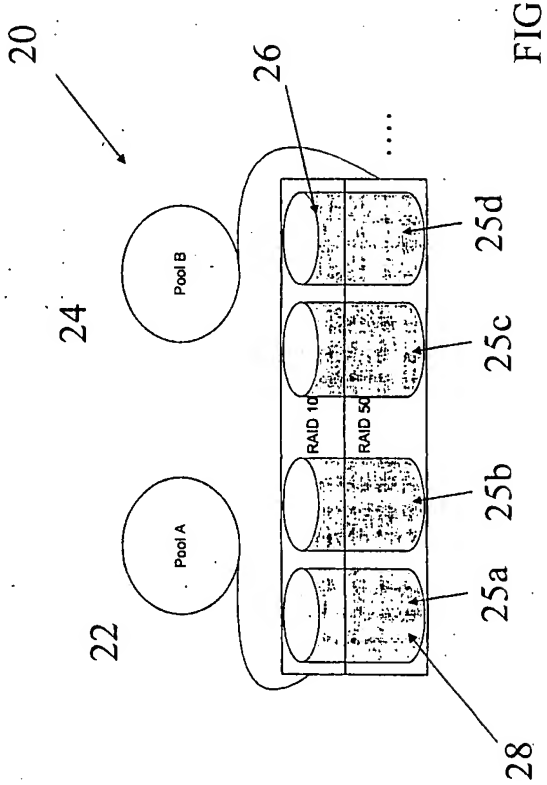


FIG. 2

From U.S. Utility Application No. 10/761,884
Filed January 20, 2004

EXHIBIT
tabbies
C

FIG. 3

From U.S. Utility Application No. 10/761,884
Filed January 20, 2004

EXHIBIT
tabbies
D

FIG. 8

From U.S. Provisional Application No. 60/441,810
Filed January 21, 2003

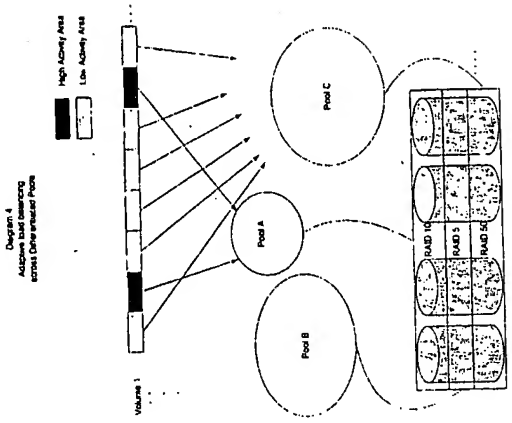


FIG. 9

From U.S. Provisional Application No. 60/441,810
Filed January 21, 2003

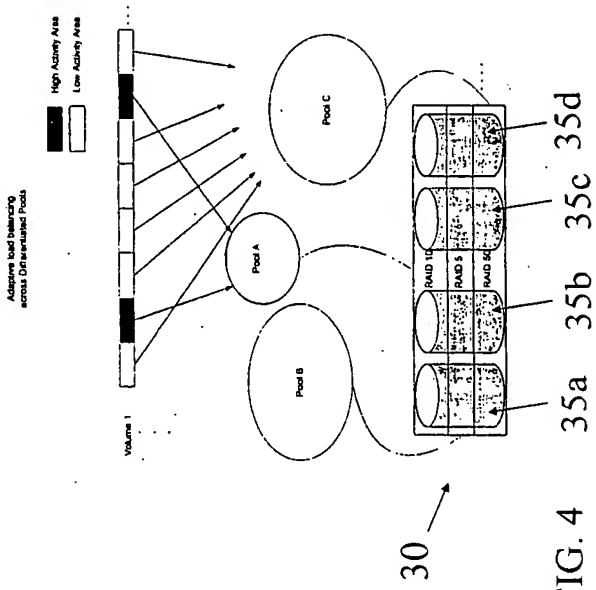


FIG. 4

From U.S. Utility Application No. 10/761,884
Filed January 20, 2004

EXHIBIT
tabbies
E

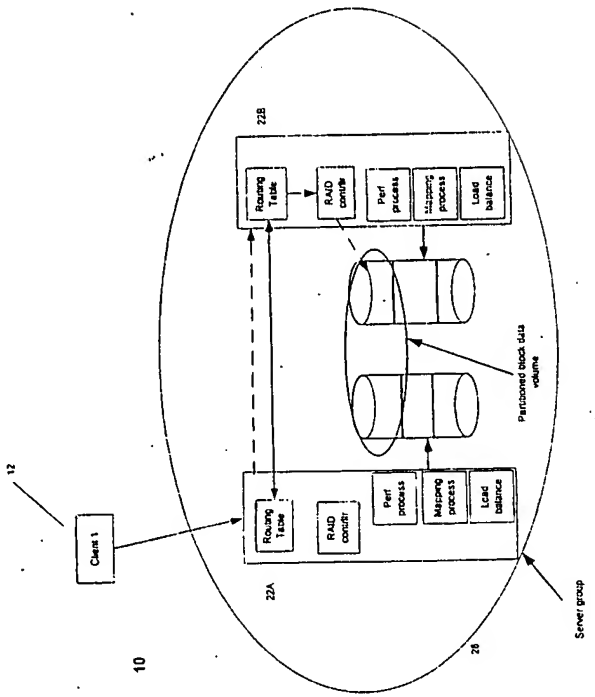


FIG 5

From U.S. Provisional Application No. 60/441,810
Filed January 21, 2003

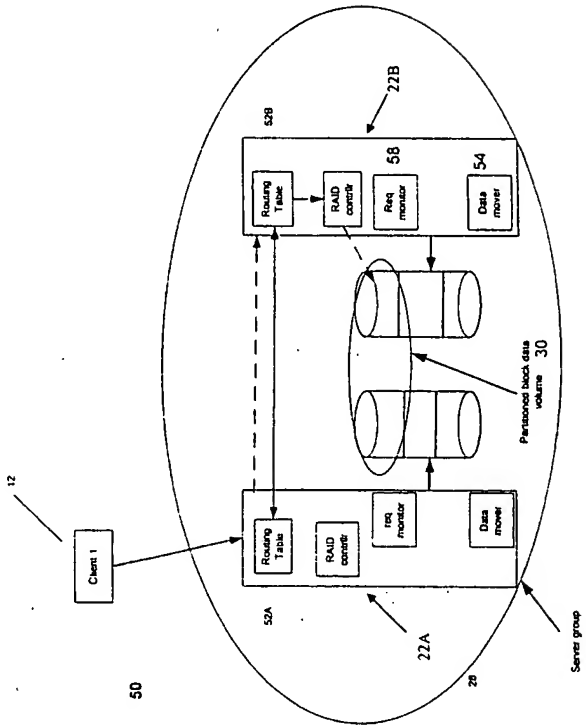


FIG 5

From U.S. Utility Application No. 10/761,884
Filed January 20, 2004

EXHIBIT
tabbies
F

From U.S. Provisional Application No. 60/441,810
Filed January 21, 2003

TITLE OF INVENTION: EQLC-PXX-005

Differentiated Storage Pools over a common set of storage devices with multiple RAID levels on each device.

TECHNOLOGY TO PROTECT:

Every patent application is required to end with at least one concise statement, called a claim, of the subject matter that the patent is to protect. To this end, please provide a concise statement of the technology (system or process) you want this patent to protect. Focus on what you want to get a patent on for the company, and do not be concerned that you are not sure it is patentable. We will check the prior art to determine patentability before preparing the application. We will use your statement regarding what you want to protect to determine whether there is an available patent position for your company.

In a block storage environment with multiple devices, differentiated pools of storage are created by applying multiple RAID techniques simultaneously on each device. These pools of storage differ in regards to performance and reliability. Adaptive load balancing can be used to place data in pools to optimize service for a single or multiple consumers of storage services.

For example:

a) Storage devices offer different performance across their LBN space: some LBN ranges have higher bandwidth for read and/or write, some LBN ranges have higher throughput for small and/or random operations, some LBN ranges have slower bandwidth, and some LBN ranges have slower throughput.
b) RAID levels have different performance characteristics for bandwidth and throughput for read and write operations, as well as in both normal mode and degraded operating modes.
c) Device performance differences across LBN space, and performance and reliability differences of RAID techniques, can be combined of to create storage pools that offer different classes of service across a common set of storage devices to the consumers of storage services.

Adaptive load balancing provides the ability to adjust data placement in pools for continuous optimal performance.

The technology to be protected is – minimally – :

1) The use of multiple raid levels concurrently on each device to create differentiated classes of storage service.
2) The use of multiple raid levels concurrently on each device in combination with device LBN performance differences to create differentiated classes of storage service.
3) Load balancing storage services across multiple differentiated classes of storage service:
    a. Automatic, adaptive load balancing of a single or multiple storage services (volumes or file systems) across pools via
        i. Volume based allocation
        ii. Extent based allocation
        iii. Page-based allocation
    b. Manual load balancing of whole volumes

DESCRIPTION OF INVENTION:

Please provide a brief description of your invention, making sure that your description is directed to the technology you want to protect as described in the section above. You can use any format you chose, but experience shows that it is most helpful to provide two or three simple drawings that illustrate the invention. These can be flow charts, functional block diagrams or suitable sketches. Examples of such drawings are provided at the end of this document. We prefer to receive the drawings in electronic form, such as PowerPoint, Visio or SmartDraw. Once the drawings are complete, please add one or two paragraphs that describe what is being shown in the drawings. Please make sure that you identify each element or step shown in the drawings. Consider including a summary of results achieved, features believed to be novel, further experimental work planned, and any additional information.

[redacted]